

# Software Tools

FOR THE PROFESSIONAL PROGRAMMER

## STRIKINGLY STRUCTURED

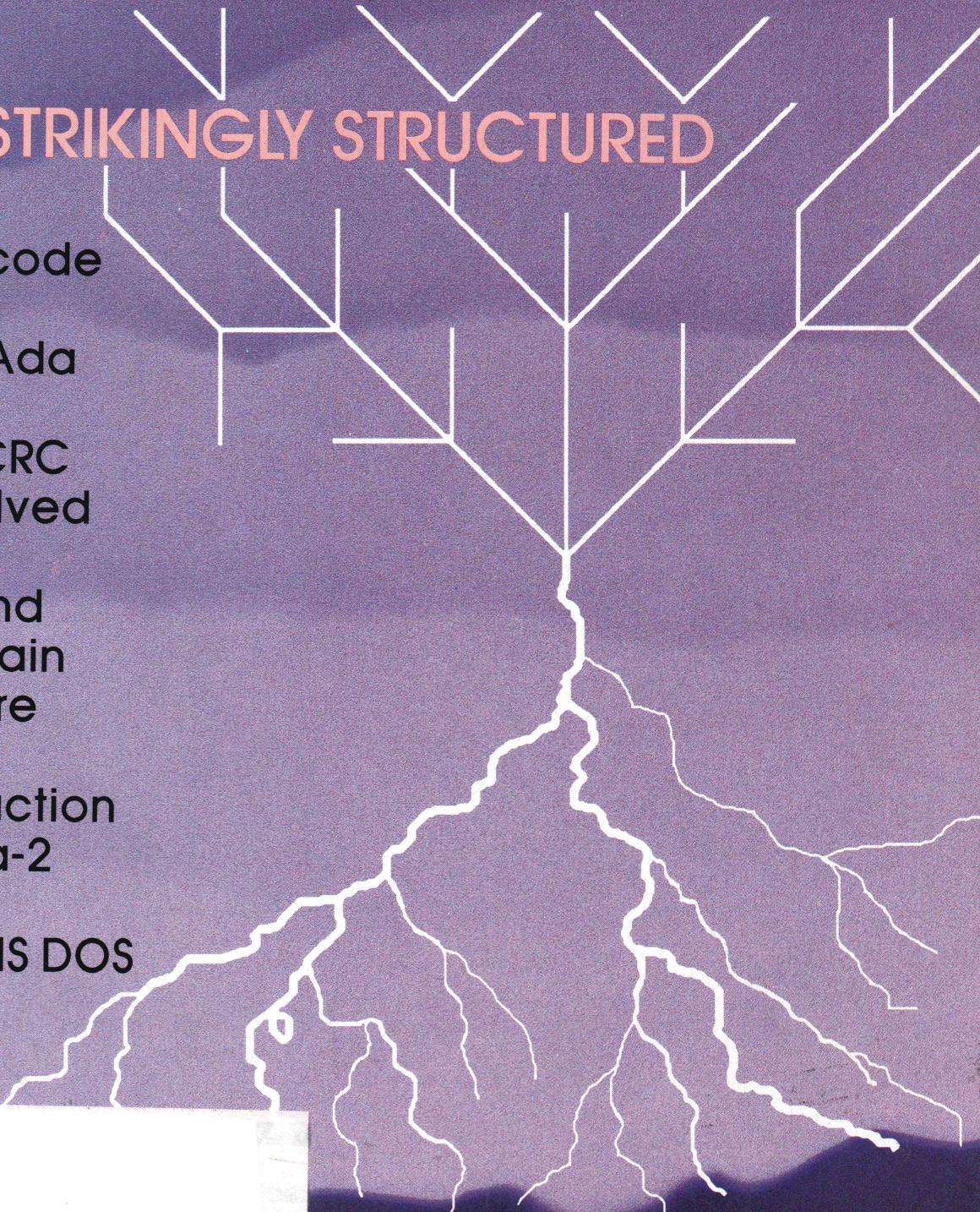
Structured code  
in Pascal,  
Modula-2, Ada

The Great CRC  
Mystery: Solved

Where to Find  
Public-Domain  
Ada Software

Data Abstraction  
with Modula-2

A Shell for MS DOS



## Breakthrough for C Programmers



# H.E.L.P. Eliminates Every Bug known to Compilers ... As well as a few other species

**H.E.L.P.** is a completely interactive C programming environment with three innovative full-sized features that will revolutionize the way you write code.

### A Clean Compile — Guaranteed!

Say Good-bye to all compiler-type errors. **H.E.L.P.**'s built-in program checker (which would embarrass LINT) not only hunts down bugs...explains the proper syntax...gives examples of usage...but will even offer suggested corrections. If you want, **H.E.L.P.** will even make the corrections for you...at the touch of a key.

**H.E.L.P.** also finds semantic errors as well as poor style and inefficiencies. You can even check the portability of your code!

### Multi-Window Editing

Open as many windows as you want...there's no limitation...not even your own memory. Because **H.E.L.P.** uses a virtual memory system you can create programs larger than your machine capacity.

**H.E.L.P.**'s very powerful editor allows you the flexibility to work in several windows...with several files at the same time.

Circle no. 165 on reader service card

### Save Keystrokes

Hundreds of commands are bound to the keyboard to give you fast execution.

### Always be in Control

Not only can you develop code in many windows at the same time, but you can show (and refer to) important definitions in one window while creating in another. Or open a window and keep notes about your program...or type a memo...or a letter.

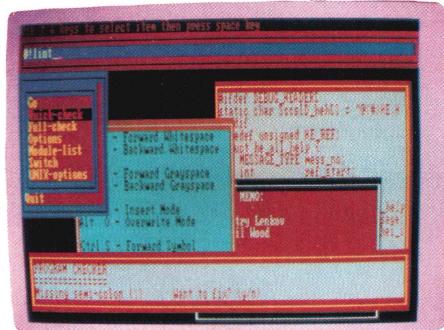
### Increase your Productivity by 300% or More . . . .

If you are a novice programmer, you'll begin writing code like an advanced programmer much faster with **H.E.L.P.**.

Just imagine what **H.E.L.P.** will do for the ADVANCED PROGRAMMER.

You'll have more time to become creative with your algorithm (since **H.E.L.P.** will make sure your code compiles the "first time at bat").

**H.E.L.P.** tracks every step you make. If you are not sure about a command, just press a key, and you'll get the kind of help you need.



### Check These Features

- Multi-window environment
- Interactive program checking
- Check syntax, semantic, type usage, intermodule inconsistencies and portability
- Multi-file editing
- Intelligent help subsystem
- User-definable keyboard bindings
- Supports color and monochrome
- **H.E.L.P.** supports the full C Language

### NOW IN MS-DOS

**EVEREST**  
SOLUTIONS

Order now \$395

Everest Solutions, Inc.  
3350 Scott Boulevard  
Building 58  
Santa Clara, CA 95051  
(408) 986-8977



Optotech, Inc.

# The 5 1/4 inch Optical Disk Drive Is Here!

## Optical Disk Drive 5984

- 200 megabytes on a removable cartridge.
- Fast access read and write.
- For PCs and minis.
- Extensive interface software.
- Available Now.

## The Preferred Solution For:

- Database—large, portable, indelible and updatable
- Online Mass Storage—integral backup
- Imaging—capacity with removability



Optotech, Inc.

770 Wooten Road  
Colorado Springs, CO 80915  
303.570.7500 Telex 592966

# Mark Williams

Now the biggest name  
in C compilers comes in a size  
everybody can afford.  
**Let's C™**

Introducing **Mark Williams' \$75 C compiler**. Want to explore C programming for the first time? Or just on your own time? Now you can do it in a big way without spending that way. With Let's C.

This is no little beginner's model. Let's C is a powerful programming tool, packed with all the essentials of the famous Mark Williams C Programming System. The one chosen by Intel, DEC, Wang and thousands of professional programmers. The one that wins the benchmarks and the reviewers' praise:

*"(This compiler) has the most professional feel of any package we tested..."*—BYTE  
*"Of all the compilers reviewed, (it) would be my first choice for product development."*—David W. Smith, PC WORLD

And now for more big news. Get our revolutionary csd C Source

Debugger for just \$75, too. You can breeze through debugging at the C source level ignoring clunky assembler code.

Affordable, powerful, debuggable. Mark Williams Let's C is the big name C compiler at a price you can handle. Get your hands on it now.

- For the IBM-PC and MS-DOS
- Fast compact code plus register variables
- Full Kernighan & Ritchie C and extensions
- Full UNIX™ compatibility and complete libraries
- Small memory model
- Many powerful utilities including linker, assembler, archiver, cc one-step compiling, egrep, pr, tail, wc
- MicroEMACS full screen editor with source
- Supported by dozens of third party libraries
- Upgradeable to C Programming System for large scale applications development

**Let's C Benchmark Done on an IBM-PC/XT, no 8087.**  
Program: Floating Point from BYTE, August, 1983.

Exec Time in Seconds

Let's C	134.20
MS 3.0	347.45

Use this coupon or charge by calling toll-free:  
1-800-MWC-1700. In Ill. call 312-472-6659.

## Mark Williams Let's C

**\$75**

Please send me:

\_\_\_\_ copies of Let's C and \_\_\_\_ copies of csd (C Source Debugger) at \$75 each. (Ill. residents add 7% sales tax.)

Check  Money Order  Visa, MasterCard or American Express

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Card # \_\_\_\_\_ Exp. Date \_\_\_\_\_

Signature \_\_\_\_\_



**Mark  
Williams  
Company**

1430 West Wrightwood  
Chicago, Illinois 60614

**Dr. Dobb's Journal of**

# Software Tools

## ARTICLES

**Why some versions of XMODEM are too slow**

**Draw Poker: an alternative to Star Wars**

**20 megabytes of public-domain Ada code**

**Living without MS DOS's command.com**

**Simulating human cognition with hardware?**

**PASCAL: The Great CRC Mystery** **26**  
by Terry Ritter

The theory and practice of the cyclic redundancy check, an error-detection technique whose misuse can lead to undetected error.

**PASCAL: Fast Integer Powers for Pascal** **36**  
by Dennis E. Hamilton

This program implements the fastest known algorithm for the computation of arbitrary integer powers.

**ADA: Learning Ada on a Micro** **42**  
by Do-While Jones

Do-White enlists us for basic training in Ada, using a problem domain familiar to most GIs.

**ADA: The DOD Ada Software Repository** **60**  
by Richard Conn

A guided tour of the Defense Data Network's 300-plus files of public-domain Ada programs. Security clearance not required.

**MODULA-2: Data Abstraction with Modula-2** **62**  
by Bill Walker and Stephen Alexander

Most languages confound the abstract definition of a data structure with the details of its implementation. The authors, in constructing a priority queue, show how to separate the two.

## COLUMNS

**C CHEST: A New Shell for MS DOS (continued)** **16**  
by Allen Holub

The history, shell variable, and alias support routines, with notes on compiling the shell and portability issues.

**16-BIT TOOLBOX: Recommended Software** **114**  
by Ray Duncan

Ray's recommendations, plus Trojan horse programs you'll want to avoid. Also: expanding the environment and more on decision variables.

## FORUM

**EDITORIAL: Evangelism** **6**  
by Michael Swaine

**LETTERS: Our readers set us straight** **8**

**CARTOON: The bugs of Rand Renfroe** **8**

**VIEWPOINT: Sixth Generation Minds** **12**  
by Richard Grigoris

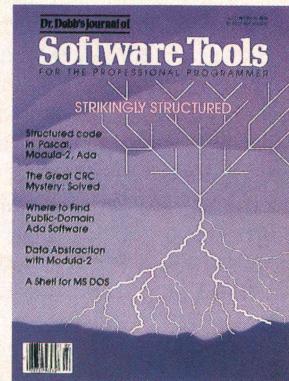
**DDJ ON LINE: What's up** **14**  
by Frank DeRose

## PROGRAMMERS' SERVICES

**PROFESSIONAL PROGRAMMER:** **121**  
Startup stories

**CHIPWATCH: Semiconductors and software** **123**

**OF INTEREST: New Products** **124**



This month's cover was created by West and Moravec Designs. The image was created with MacPaint and printed on a LaserWriter. The background was created with an airbrush.

**This Issue**

Pascal, Ada, and Modula-2 are regarded as languages that encourage structured programming. This issue presents for each language some code that we hope you will find educational, useful, or both. We found Terry Ritter's explication of CRCs particularly enlightening.

**Next Issue**

For forty years, the fundamental algorithms underlying the design of computer software and hardware have been, with a few controlled exceptions, sequential. Now, even users are coming to see the benefits of background tasking and print spooling, while multiprocessor architectures are approaching practicality. But application concurrency and multiprocessor designs will only scratch the surface of parallelism until we recast algorithms in parallel. Is some new Knuth even now writing the book on parallel algorithms?

# This is for all the power users technologies before they

Other than Steve Wozniak and Jonathan Rotenberg, there are probably only 2,998 personal computer users who qualify as trend setters. They're the people who owned Apples® when everyone else thought Apple® was a record label. People who were called hackers when a hacker was someone no one wanted to play golf with.

However many of you there are, this ad is for you. It's been designed and written to introduce you to a new technology without using superlatives or words like revolutionary. (We're saving those words for future ads targeted at the general consumer.)

The new technology is called the Softstrip™ System. This ad tells you what it's all about.

## THE SOFTSTRIP SYSTEM ENCODES DATA ONTO PAPER.

Softstrip technology allows text, graphics, even digitized sound to be encoded on a strip of paper. Providing an alternative to magnetic media and telecommunications for the recording, distribution and retrieving of information.

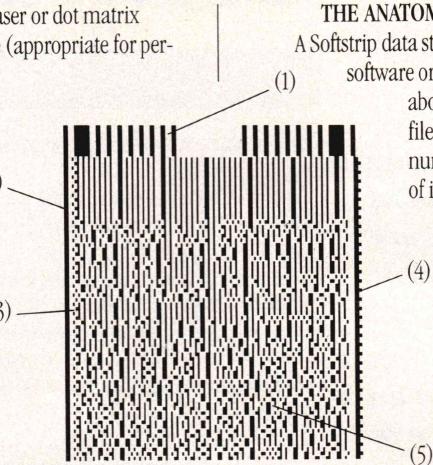
These data strips, each a structured pattern of black and white rectangles that look something like a condensed bar code, can be encoded with special software and read with a scanning device called the Cauzin Softstrip System Reader. The reader optically scans the strip, translates its contents into 8-bit code and feeds it into a personal computer's serial or cassette port, enabling automatic, error-free entry of printed data without using a keyboard.

publishers), or by using a laser or dot matrix printer and special software (appropriate for personal or business use).

If you want, you can generate strips that can be reproduced on a copier or versions that can't be. Either way, any data strip, whether it's printed in a newspaper, magazine or personal letterhead, can survive pen marks, scratches, even coffee stains.

Basically, anything you can put on a magnetic disk you can put on a Softstrip data strip, which should suggest numerous application possibilities.

Starting in the next two months, data strips will appear in magazines, journals and books. These strips will contain program listings, tables of

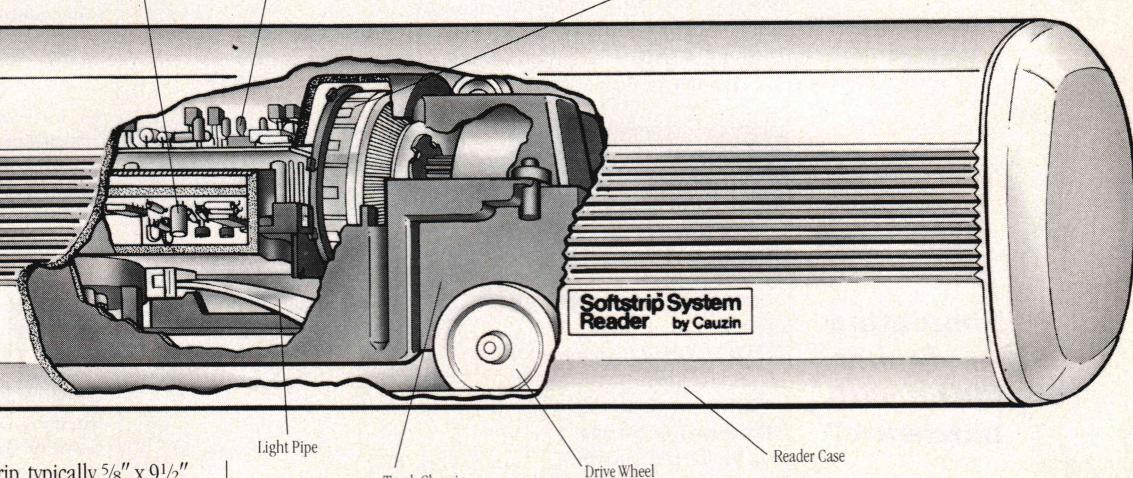
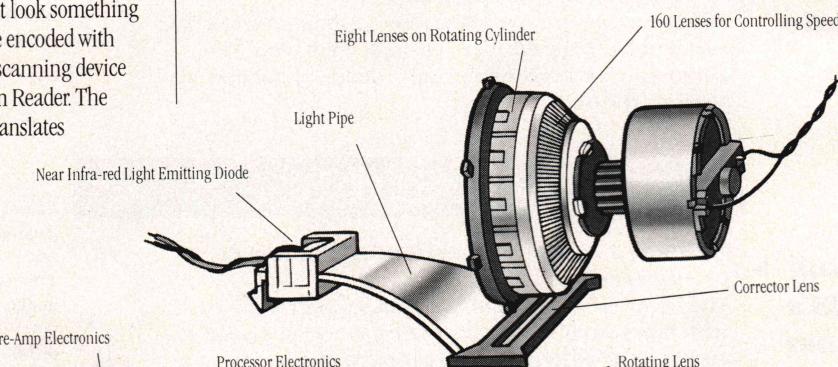


## THE ANATOMY OF A STRIP.

A Softstrip data strip contains not only software or data, but also information about its content, including file types, file name and the number of strips. Because of its inherent technology, strips are as accurate as any magnetic medium. And they can be entered into an IBM PC®, Apple II® or Macintosh® computer using the same reader with only slightly modified communications software.

Strips encode data bit by bit using highly structured optical patterns. The bits are each composed of two rectangles called di-bits. They function as optical on/off signals. White/black equals 1. Black/white equals 0.

Data is organized in lines. Each line, between 0.01 and 0.04 inches high and from 0.5 to 0.76 inches wide, contains from two to six bytes of data. Line width and height are varied depending upon the quality of the paper and printing process. The reader



One Softstrip data strip, typically  $5/8'' \times 9\frac{1}{2}''$ , can contain up to 5,500 bytes of information. (For example, you could fit this entire ad on two strips.)

Strips can be printed by using a photographic negative (ideal for book and magazine

contents, new product demonstrations and data.

Eventually, strips may be used for everything from bank statements to sheet music.

scans each data line with a series of raster scans 0.0025 inches apart providing between four and sixteen scans per line.

# who want to know about new become household words.

A close-up view of a strip reveals five distinct sections. The header (1) at the top tells the reader the number of bytes in a line, the height of each line, and the paper to ink contrast level. Running vertically down the sides of the strip are the startline (2), the checkerboard (3) and the rack (4). They identify the boundaries of every horizontal line to be read. They also work in tandem to feed the reader alignment information.

Contained within the body of the strip, between the checkerboard and rack, is the file's data area (5).

Strip data accuracy is checked and error correction is provided by parity bits at the beginning and end of every data line, as well as by a strip checksum. There is also an optional 16 bit CRC. Combined, this design results in an undetected bit error rate

of less than one bit error per 10,000,000,000 bits.

## IT TOOK GUTS TO BUILD THE READER.

Rated for 25,000 reads, the reader is an equally impressive technology. It's composed of two key components: the case and the truck. While the case sits still, the truck moves uniformly down the length of the strip making a complete scan of the strip's di-bit lines every 0.0025 inches.

As the truck moves down the strip, it tracks its own lateral movement within five microns. Alignment is controlled by two servo mechanisms. As the truck moves, it illuminates the area to be scanned using near infra-red light beamed through a light pipe. (The infra-red technique permits the reader to see through colors, stains, and spills.)

The reader's optical scanning system, containing eight rotating cylindrical lenses and an aspherical corrector lens, forms an F1.2 optical system with a depth of field between 0.05 and 0.08 inches. A set of 160 additional cylindrical lenses on the rotating lens allow the system to control scanning speed.

Inside the reader, the mechanical system uses six AGMA-7 high precision plastic molded gears to provide very accurate truck movement. One gear system even allows for a 4000 to 1 angle reduction with no backlash for corrector lens alignment.

A TMS 7040 8-bit processor and Cauzin's own custom VLSI chip provide

reader logic, control and communications using four nested phase locked loops and several hardware and software servos. The reader transmits data to the host at 4800 baud burst rates with throughput of 1500 baud.

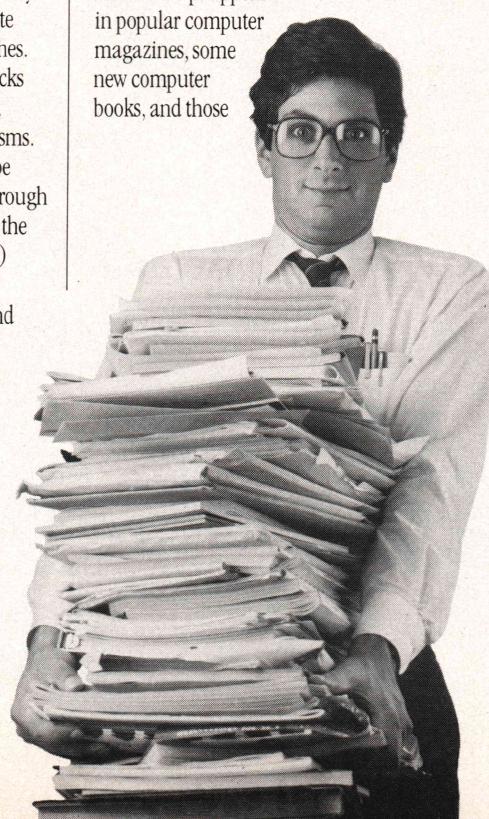
## HOW TO CREATE YOUR OWN STRIPS.

There are two ways to create Softstrip™ data strips. For large volume and greater density — up to 5500 bytes per strip — a film negative is created using special Cauzin software and hardware. This is ideal for book, magazine, newsletter, data base and commercial software publishers who can reproduce a strip in volume using web, offset, gravure or similar processes.

For personal or business applications, 500 to 1000 byte strips can be

generated using Cauzin licensed software on dot matrix printers; up to 3400 byte strips can be generated using other Cauzin licensed software and laser printers.

In the next few months, you should start to see data strips appear in popular computer magazines, some new computer books, and those



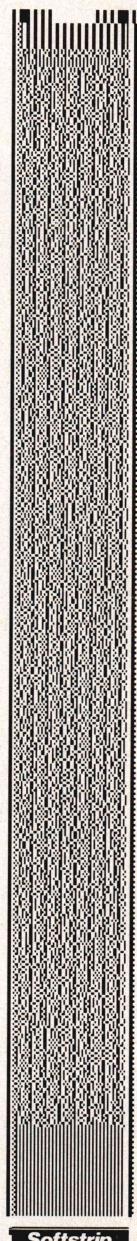
consumer ads we told you about earlier. They'll look exactly like the working strip you see here, a medium density strip with an ASCII text file on it.

Of course, you'll be able to purchase a reader at most computer dealers. They'll be selling for about \$200.00. Contact your dealer soon for a demonstration. Or call us directly at 203-573-0150.

Apple® and Macintosh® are registered trademarks of Apple Computer Inc.

Apple® is a registered trademark of Apple Records, Inc.

Softstrip® and the Softstrip® System Reader are trademarks of Cauzin Systems, Inc.



## Softstrip®

COMPUTER READABLE PRINT

Cauzin Systems, Inc.,  
835 South Main St., Waterbury, CT 06706

Circle no. 226 on reader service card.

# EDITORIAL

It's December 23 as I write this, and I'm hurrying to get this note to you before skipping out for the holidays.

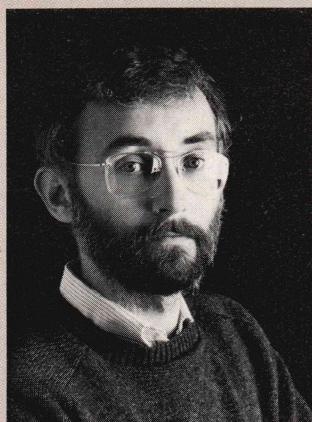
When you read this, we expect to be in the middle of moving into our spacious new offices at 501 Galveston Dr., Redwood City, CA 94063. You should send any correspondence there.

Our editorial calendar (see below) indicates what topics we intend to go after this year. But the calendar doesn't tell the whole story. It doesn't include, for example, coverage of networking issues, cryptography, and memory-resident standards, all of which we'll be looking into in 1986, nor does it mention other topics that you will surely think of. Send us your ideas.

Allen Holub wanted to pass along this suggestion: *DDJ* writers and columnists are often more than happy to respond to letters from readers. Your chances of getting a response are increased if you include a self-addressed, stamped envelope. You can also reach our columnists via our Electronic Edition.

Frank DeRose, our special projects editor, is coordinating the editorial aspects of our new book line and of the Electronic Edition of *DDJ* on CompuServe. Frank spells out in this issue what we intend to do with our online publishing experiment. When you read this, some details of implementation will probably be different from what he says here, writing as he was before Christmas. Print has its limitations, which is why we're exploring electronic publishing. Log on to find out what's really happening there.

We're also doing the groundwork for an ambitious software review program. You won't see the results of this until midyear, but last August's C review and the review of program-



mable editors in November indicate the direction we'll be taking: comparative reviews of many products; more than one programmer evaluating the products; using the most useful and objective sets of benchmarks we can coax and coerce veteran programmers into developing for us; and regular updates and bug reports on the reviews themselves. We've wisely promoted Sara Noah Ruddy (our erstwhile editorial assistant) to assistant editor with responsibility for coordinating this whole review process.

The other member of the editorial staff whose name you should know is Vince Leone, our managing editor, who puts the pieces together every month. Vince, Sara, Frank, Allen, and I will probably be joined by one other editor in the next month or two. I'll let you know.

One final note: we owe you an article. In January, we promised a look at a program that ports between dialects of Pascal. This month was packed, and we didn't have room for everything; watch for it in March.

## 1986 editorial calendar

March: The future of programming; Article deadline: past.  
April: AI. Deadline: past.  
May: Designing usable software. Deadline: past.  
June: Communications. Deadline: 3/1/86.  
July: Forth. Deadline: 4/1/86.  
August: C. Deadline: 5/1/86.  
September: Algorithms. Deadline: 6/1/86.  
October: 80286/80386 programming. Deadline: 7/1/86.  
November: Graphics. Deadline: 8/1/86.  
December: Operating systems. Deadline: 9/1/86.

A handwritten signature in cursive script that reads "Michael Swaine".

Michael Swaine

## Dr. Dobb's Journal of

# Software Tools

### Editorial

**Editor-in-Chief** Michael Swaine

**Managing Editor** Vince Leone

**Editorial Assistant** Sara Noah Ruddy

**Technical Editor** Allen Holub

**Contributing Editors** Ray Duncan

Allen Holub

**Copy Editors** Laura Kenney

Rhoda Simmons

**Special Projects Editor** Frank DeRose

### Production

**Production Manager** Bob Wynne

**Art Director** Shelley Rae Doeden

**Production Assistant** Alida Hinton

**Typesetter** Jean Aring

**Cover Artist** Randy Moravec

### Circulation

**Fulf./Newsstand Mgr.** Stephanie Barber

**Subscription Mgr.** Maureen Kaminski

**Book Marketing Mgr.** Jane Sharninghouse

**Circulation Assistant** Kathleen Shay

### Administration

**Finance Manager** Sandra Dunie

**Business Manager** Betty Trickett

**Accounts Payable Supv.** Mayda Lopez-Quintana

**Accounts Payable Assts.** Denise Giannini

Kathy Robinson

**Billing Coordinator** Laura Di Lazzaro

**Accountant** Marilyn Henry

**Adm. Coordinator** Kobi Morgan

### Advertising

**Advertising Director** Shawn Horst (415) 424-0600

### Advertising Sales

Walter Andrzejewski (617) 567-8361

Lisa Boudreau (415) 424-0600

Beth Dudas (714) 643-9439

Michael Beaty (317) 875-8093

**Systems Manager** Ron Copeland

**Administrative Manager** Anna Kittleson

**Advertising Secretary** Michelle A Davie

## M&T Publishing, Inc.

**Chairman of the Board** Otmar Weber

**Director** C.F. von Quadrat

**President and Publisher** Laird Foshay

**Dr. Dobb's Journal** (USPS 3076900 is published monthly by M&T Publishing, Inc., 2464 Embarcadero Way, Palo Alto, CA 94303, (415) 424-0600, Second class postage paid at Palo Alto and at additional entry points.

Address correction requested. Postmaster: Send Form 3579 to **Dr. Dobb's Journal**, P.O. Box 27809, San Diego, CA 92128.

**ISSN 0884-5395**

**Customer Service:** For subscription problems call: outside CA 800-321-3333; within CA 619-485-9623 or 566-6974. For book, back issue, or disk order problems call 415-424-1474.

**Subscription Rates:** \$29.97 per year within the United States. Foreign subscription rates: \$56.97 for airmail, \$46.97 for surface mail. Foreign subscriptions must be prepaid in U.S. Dollars, drawn on a U.S. Bank.

**Foreign Distributor:** Worldwide Media Service, Inc., 386 Park Ave. South, New York, NY 10016, (212) 686-1520 TEL-EX: 620430 (WUI)

Entire contents copyright © 1986 by M&T Publishing, Inc. unless otherwise noted on specific articles. All rights reserved.



## People's Computer Company

**Dr. Dobb's Journal** is published by M&T Publishing, Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a non-profit, educational corporation.



# The C for Microcomputers

PC-DOS, MS-DOS, CP/M-86, Macintosh, Amiga, Apple II, CP/M-80, Radio Shack, Commodore, XENIX, ROM, and Cross Development systems

## MS-DOS, PC-DOS, CP/M-86, XENIX, 8086/80x86 ROM

### Manx Aztec C86

*"A compiler that has many strengths... quite valuable for serious work."*

Computer Language review, February 1985

**Great Code:** Manx Aztec C86 generates fast executing compact code. The benchmark results below are from a study conducted by Manx. The Dhystone benchmark (CACM 10/84 27:10 p108) measures performance for a systems software instruction mix. The results are without register variables. With register variables, Manx, Microsoft, and Mark Williams run proportionately faster. Lattice and Computer Innovations show no improvement.

	Execution Time	Code Size	Compile/Link Time
<b>Dhystone Benchmark</b>			
Manx Aztec C86 3.3	34 secs	5,760	93 secs
Microsoft C 3.0	34 secs	7,146	119 secs
Optimized C86 2.20J	53 secs	11,009	172 secs
Mark Williams 2.0	56 secs	12,980	113 secs
Lattice 2.14	89 secs	20,404	117 secs

**Great Features:** Manx Aztec C86 is bundled with a powerful array of well documented productivity tools, library routines and features.

Optimized C compiler	Symbolic Debugger
AS86 Macro Assembler	LN86 Overlay Linker
80186/80286 Support	Librarian
8087/80287 Sensing Lib	Profiler
Extensive UNIX Library	DOS, Screen, & Graphics Lib
Large Memory Model	Intel Object Option
Z (vi) Source Editor -c	CP/M-86 Library -c
ROM Support Package -c	INTEL HEX Utility -c
Library Source Code -c	Mixed memory models -c
MAKE, DIFF, and GREP -c	Source Debugger -c
One year of updates -c	CP/M-86 Library -c

Manx offers two commercial development systems, Aztec C86-c and Aztec C86-d. Items marked -c are special features of the Aztec C86-c system.

<b>Aztec C86-c Commercial System</b>	<b>\$499</b>
<b>Aztec C86-d Developer's System</b>	<b>\$299</b>
<b>Aztec C86-p Personal System</b>	<b>\$199</b>
<b>C-tree database (source)</b>	<b>\$399</b>

All systems are upgradable by paying the difference in price plus \$10.

**Third Party Software:** There are a number of high quality support packages for Manx Aztec C86 for screen management, graphics, database management, and software development.

<b>C-tree \$395</b>	<b>Greenleaf \$185</b>
<b>PHACT \$250</b>	<b>PC-lint \$98</b>
<b>HALO \$250</b>	<b>Amber Windows \$59</b>
<b>PRE-C \$395</b>	<b>Windows for C \$195</b>
<b>WindScreen \$149</b>	<b>FirsTime \$295</b>
<b>SunScreen \$99</b>	<b>C Util Lib \$185</b>
<b>PANEL \$295</b>	<b>Plink-86 \$395</b>

## MACINTOSH, AMIGA, XENIX, CP/M-68K, 68k ROM

### Manx Aztec C68k

*"Library handling is very flexible... documentation is excellent... the shell a pleasure to work in... blows away the competition for pure compile speed... an excellent effort."*

Computer Language review, April 1985

Aztec C68k is the most widely used commercial C compiler for the Macintosh. Its quality, performance, and completeness place Manx Aztec C68k in a position beyond comparison. It is available in several upgradable versions.

Optimized C	Creates Clickable Applications
Macro Assembler	Mouse Enhanced SHELL
Overlay Linker	Easy Access to Mac Toolbox
Resource Compiler	UNIX Library Functions
Debuggers	Terminal Emulator (Source)
Librarian	Clear Detailed Documentation
Source Editor	C-Stuff Library
MacRAM Disk -c	UniTools (vi,make,diff,grep) -c
Library Source -c	One Year of Updates -c

Items marked -c are available only in the Manx Aztec C86-c system. Other features are in both the Aztec C86-d and Aztec C68k systems.

### Aztec C68k-c Commercial System

**\$499**

### Aztec C68d-d Developer's System

**\$299**

### Aztec C68k-p Personal System

**\$199**

### C-tree database (source)

**\$399**

### AMIGA, CP/M-68k, 68k UNIX

**call**

## Apple II, Commodore, 65xx, 65C02 ROM

### Manx Aztec C65

*"The AZTEC C system is one of the finest software packages I have seen."*

NIBBLE review, July 1984

A vast amount of business, consumer, and educational software is implemented in Manx Aztec C65. The quality and comprehensiveness of this system is competitive with 16 bit C systems. The system includes a full optimized C compiler, 6502 assembler, linkage editor, UNIX library, screen and graphics libraries, shell, and much more. The Apple II version runs under DOS 3.3, and ProDOS. Cross versions are available.

The Aztec C65-c/128 Commodore system runs under the C128 CP/M environment and generates programs for the C64, C128, and CP/M environments. Call for prices and availability of Apprentice, Personal and Developer versions for the Commodore 64 and 128 machines.

### Aztec C65-c ProDOS & DOS 3.3

**\$399**

### Aztec C65-d Apple DOS 3.3

**\$199**

### Aztec C65-p Apple Personal System

**\$99**

### Aztec C65-a for learning C

**\$49**

### Aztec C65-c/128 C64, C128, CP/M

**\$399**

### Distribution of Manx Aztec C

In the USA, Manx Software Systems is the sole and exclusive distributor of Aztec C. Any telephone or mail order sales other than through Manx are unauthorized.

## Manx Cross Development Systems

Cross developed programs are edited, compiled, assembled, and linked on one machine (the HOST) and transferred to another machine (the TARGET) for execution. This method is useful where the target machine is slower or more limited than the HOST. Manx cross compilers are used heavily to develop software for business, consumer, scientific, industrial, research, and educational applications.

**HOSTS:** MS-DOS, CP/M-86, Macintosh, CP/M-68k, CP/M-80, TRS-80 3 & 4, Apple II, Commodore C64, 8086/80x86 ROM, 68xxx ROM, 8080/8085/Z80 ROM, 65xx ROM.

The first TARGET is included in the price of the HOST system. Additional TARGETS are \$300 to \$500 (non VAX) or \$1000 (VAX).

Call Manx for information on cross development to the 68000, 68816, Amiga, C128, CP/M-68K, VRTX, and others.

## CP/M, Radio Shack, 8080/8085/Z80 ROM

### Manx Aztec CII

*"I've had a lot of experience with different C compilers, but the Aztec C80 Compiler and Professional Development System is the best I've seen."*

80-Micro, December, 1984, John B. Harrell III

### Aztec C II-c (CP/M & ROM)

**\$349**

### Aztec C II-d (CP/M)

**\$199**

### C-tree database (source)

**\$399**

### Aztec C80-c (TRS-80 3 & 4)

**\$299**

### Aztec C80-d (TRS-80 3 & 4)

**\$199**

### How To Become an Aztec C User

To become an Aztec C user call 1-800-221-0440 or call 1-800-832-9273 (800-TEC WARE). In NJ or outside the USA call 201-530-7997. Orders can also be telexed to 4995812.

Payment can be by check, COD, American Express, VISA, Master Card, or Net 30 to qualified customers.

Orders can also be mailed to Manx Software Systems, Box 55, Shrewsbury, NJ 07701.

### How To Get More Information

To get more information on Manx Aztec C and related products, call 1-800-221-0440, or 201-530-7997, or write to Manx Software Systems.

### 30 Day Guarantee

Any Manx Aztec C development system can be returned within 30 days for a refund if it fails to meet your needs. The only restrictions are that the original purchase must be directly from Manx, shipped within the USA, and the package must be in resalable condition. Returned items must be received by Manx within 30 days. A small restocking fee may be required.

### Discounts

There are special discounts available to professors, students, and consultants. A discount is also available on a "trade-in" basis for users of competing systems. Call for information.

To order or for information call:

**800-221-0440**

Circle no. 108 on reader service card.

## LETTERS



## Columns

Dear DDJ,  
Allen Holub's C Chest article on recursive descent parsing (September 1985) was very interesting but contained an inaccurate grammar that gives erroneous results such as:

$2*3+1 = 8$   
(should be evaluated as  $(2*3)+1 = 7$ )

$6-5-4 = 5$   
(should be evaluated as  $(6-5)-4 = -3$ )

A correct grammar for expressions is found, in one form or another, in every book on compiling. I will show how the rather unintuitive final form is reached by proceeding in small steps.

A very simple grammar is given by the following rules:

$E ::= E A E | (E) | -E | n$   
 $A ::= + | - | * | /$

where E and n stand for expression and number respectively, and A stands for arithmetic operator. This grammar is ambiguous, as  $6-5-4$  can be parsed in two valid (but different) ways:

$(6-5)-(4)$   
 $(6)-(5-4)$

In each case, the parenthesised symbols form an expression, and the two expressions are linked by an

arithmetic operator. We need a rule to introduce associativity, i.e., to ensure the first parse of  $6-5-4$ . Rule (5) in Figure 6 of the article attempts to do so but introduces the wrong associativity:

$E ::= F-E$

forces the incorrect parse

$6-5-4=6-(5-4)$

Inverting the order in rules (2) to (5) will solve the associativity problem. This leaves us with a second problem. We want both  $2*3+1$  and  $1+2*3$  to evaluate as 7. This is because in mathematics multiplication and division have a higher precedence than addition and subtraction. To achieve this we introduce two new nonterminals, *term* and *factor*. Roughly speaking, terms are things that get added (or by extension subtracted) together, while factors get multiplied (or divided). The resulting grammar is:

$E ::= E+T | E-T | T$   
 $T ::= T*F | T/F | F$   
 $F ::= n | -n | (E) | -(E)$

This grammar is correct. It will not parse  $2*3+1$  as  $(2)*(3+1)$  because that would be an expression multiplied by an expression, not one of the valid ways of forming an expression.

Though correct, this form is not suitable for parsing by recursive descent. There is no way to "get at" the first piece to start the recursion. The grammar given by Holub allowed us to start *expr()* with (to evaluate  $6-5-4$ ):

$lval = factor();$   
 $lval -= expr();$

Our grammar would seem to require:

$lval = expr();$   
 $lval -= factor();$

So *expr()* would call itself in an infinite loop. The grammar must therefore be modified as follows:

$E ::= TE'$   
 $E' ::= +TE' | -TE' | NULL$   
 $T ::= FT'$   
 $T' ::= *FT' | /FT' | NULL$   
 $F ::= n | -n | (E) | -(E)$

$E'$  and  $T'$  are the new nonterminals required to make the basic grammar parsable by recursive descent. They allow us to split off a piece to get started.

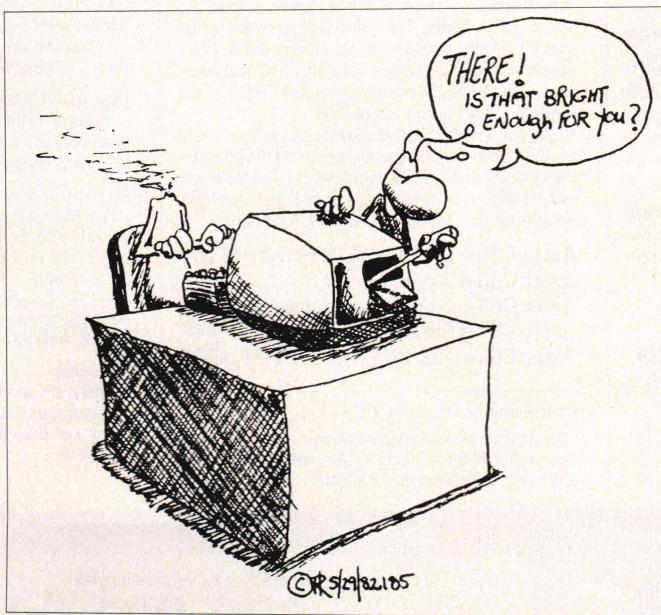
Note that the definition of a factor is unchanged, so the code for *factor()* and *constant()* will not need to be modified.

Mohamed el Lozy  
Health Sciences Computing Facility  
665 Huntington Ave.  
Boston, MA 02115

## DISnDATA

Dear DDJ,  
In your November 1985 issue on page 89, you ran an ad for a disassembler written by C. W. Medlock of PRO/AM SOFTWARE. On the basis of claims made in the ad, I purchased this product for the price of \$145.

To my sorrow, when I received this software and tried it, I found that, for the vast majority of programs, it does not correctly locate the data and code areas as claimed in the ad. Instead, it invariably outputs large segments of code areas as data. What is worse is that there are no switches in the program to override the automatic "algorithm" when this occurs, nor is there any provision for telling the program where the data and code areas are if, as is usually the case, you happen to know this information. The only recourse you have, as described in the 20-page manual supplied with the program, is to do a series of binary



SOURCE CODE  
INCLUDED!

# Multiuser, Multitasking Mainframe Performance from your PC, XT or AT . . .

... With WENDIN®  
Operating Systems  
and Software  
Development Tools.

## Operating System Toolbox™ only \$9900

Operating System Toolbox™ is a software construction set that enables you to build your own multitasking, multiuser operating system. All you need is your creativity and imagination to write a shell and link it with the Toolbox. Your Operating System will support your own commands, WENDIN standard system services and your special user interface. Your Operating System will run most MS-DOS and PC-DOS programs without modification. Includes full source code written in Microsoft "C", organized into seven basic modules, all on disk. You also get object modules (in case you don't need to compile the whole Kernel), plus a fantastic in-depth, step-by-step instruction manual on how to build your personal operating system.

The best part of the news is the price — so low, you'll want to order today, just to keep up with the state of the art in operating systems.

ORDER HOTLINE  
**509/235-8088**  
Master Card & Visa



Foreign orders inquire about shipping.  
Domestic orders add \$3.50/1st item, \$1.00 each additional item for shipping, handling and insurance.

Washington residents add 7.8% sales tax.

**WENDIN**®

BOX 266  
CHENEY, WA 99004

The people who make quality  
software tools affordable.

DEALER INQUIRIES WELCOME!

PC-DOS is a trademark of IBM  
MS is a trademark of Microsoft  
Unix is a trademark of AT&T  
VAX/VMS is a trademark of Digital Equipment Corporation

Wendin and XTC are Registered Trademarks of Wendin, Inc.

PCVMS, PCUNIX, and Operating System Toolbox are Trademarks of Wendin, Inc.

## PCUNIX™ only \$9900

PCUNIX™ is a true multitasking, multiuser operating system similar to the popular UNIX® operating system for AT&T, selling for thousands of dollars more. It includes nearly 50 utilities, designed to make your program development a snap.

PCUNIX™ runs most MS-DOS and PC-DOS programs, so that you'll never have to buy another set of development tools! You can use existing compilers, linkers and editors. Programs run under PCUNIX™ can use MS-DOS system calls plus over 70 enhanced system services found in our Operating System Toolbox™. You get 4 DSDD disks jam packed with the source code to the shell and all utilities hand crafted in Microsoft "C".

The source code to the Kernel comes with Operating System Toolbox™, sold separately.

## PCVMS™ only \$9900

PCVMS™ is Wendin's version of the popular VAX/VMS operating system, which was developed by Digital Equipment Corporation for their line of VAX computers. PCVMS™ turns your IBM-PC, XT, AT or true compatible into a supercharged, multitasking, multiuser workstation that runs MS-DOS and PC-DOS programs. Programs run under PCVMS™ can use MS-DOS system calls plus over 70 enhanced system services found in our Operating System Toolbox™.

PCVMS™ comes with full source code to the PCVMS™ shell and its utilities, hand crafted in "C", and supports up to 3 users using additional serial terminals.

The source code to the Kernel comes with Operating System Toolbox™, sold separately.

## XTC® only \$9900

XTC® is the world's first multitasking editor for the IBM-PC. It also runs on IBM-XT and IBM-AT computers, as well as true compatibles. Designed BY programmers FOR programmers, XTC is the ultimate editing tool for software developers using C, PASCAL, ASSEMBLY, BASIC, FORTRAN, and other languages.

Why is XTC® the ultimate tool for editing in YOUR development environment? Because it has powerful features like MULTITASKING MACROS, WINDOWS, TEXT BUFFERS, UNTO N TIMES, MACRO PROGRAMMING CONTROL STRUCTURES and VARIABLES, as well as blinding speed that leaves other editors in the stone age.

XTC® comes with full source code written in Microsoft Pascal, on 3 DSDD disks.

## LETTERS:

(Continued from page 8)

patches to the executable file for each area of code the program doesn't find. Needless to say this is a lengthy and time-consuming process. In addition, I found that for some input programs the disassembler hangs the computer so that a reboot is necessary.

Gerald Coquin  
132 Rotary Dr.  
Summit, NJ 07901

## Of Interest

Dear *DDJ*,  
In reading through the November 1985 issue of *Dr. Dobb's Journal*, I ran across a notice concerning our C-Link product. We are pleased, of course, to be mentioned in your publication; however, the information in Alex Ragen's Of Interest column is out of date.

During the past few months since the last news release was issued, we have changed our market strategy in response to the lack of Unix and C experience among users. The main focus of our marketing effort is to use C-Link as a tool to provide a translation service. We will continue to offer C-Link as a product but only to users willing to be trained in its usage. When sold in this manner, the cost is \$4,995 plus \$195 for additional run-time licenses.

Please contact me directly if you would like more information about either the C-Link product or service.

James R. Getzinger  
Software Manufacturers Inc.  
20720 S. Leapwood Ave.  
Carson, CA 90746

## Fgrep

Dear *DDJ*,  
I wish to express my appre-

ciation for the excellent article by Ian Ashdown in the September 1985 issue covering FGREPC. This article has been very enlightening to me in actually applying the algorithms to construct finite state automata as outlined in the Aho and Ullman book *Principles of Compiler Design* and referred to by Mr. Ashdown in the discussion of the FGREPC program.

I have adapted the Unix C program for the BDS compiler and library for my Z80 machine and in this process have encountered two items in the source program that I wish to comment upon—they may be important because of their possible nonportable nature.

First, I would like to take exception with Mr. Ashdown's use of the definition of TRUE as  $(-1)$ . I suggest that this usage is not in the "spirit of C" and is out of context with the clearly defined usage of relational and logical operator returns in C, which K & R guarantees to be 1 for TRUE and 0 for FALSE when making explicit comparisons. Mr. Ashdown's use of  $(-1)$  may be a legacy of assembly-language programmers, and its use in the program on a stand-alone basis poses no problems, but it certainly plays havoc if other standard headers are used and TRUE gets redefined. For levity and con-

dign punishment, I suggest that Mr. Ashdown compile and run the program shown in Table 1, below.

Second, I had a problem with the *stoupper()* routine in that the resulting string lost its first letter in being put to uppercase. It is suggested that this routine is nonportable and compiler-dependent. The problem stems from the intuitive use of post-increment on the left side of an assignment expression, expecting the assignment to be made before the post-increment. The strict constructionist interpretation of operator precedence has the post-decrement operator higher than assignment and has the compiler reduce the left side of the expression below before the assignment operation is performed:

```
*temp++ =  
    toupper(*temp);
```

This results in a loss of a character. The order of evaluation is not defined by K & R in this case, and few books on C discuss these sorts of statements wherein the order of evaluation is not defined. These are pernicious when they seem so intuitive. Other cases, such as:

```
array[x] = ++x;
```

which is not defined is, I suggest, easier to recognize

```
#define TRUE 1  
#define FALSE 0  
main()  
{  
    if(TRUE == !FALSE)  
        puts("\nAll's right with the world.");  
    else  
        puts("\nThe world is upside down.");  
}
```

Table 1

intuitively and less likely to be applied by a C programmer.

It may be of use for *DDJ* to list, for example, these cases of evaluation-sensitive expressions that are not defined in K & R and lead to nonportability.

Justin Farnsworth  
65 rue Chauveau  
92200 Neuilly, France

## Unix

Dear *DDJ*,  
I am amazed at the number of products, advertised in your magazine and elsewhere, that are designed to make MS DOS look more like Unix. So far, I've seen a variety of text editors made famous under Unix, a plethora of C compilers, a word processing package that bears strange similarities to *nroff*, and an implementation of *make*. There is reportedly a package that implements the Bourne shell and comes with a variety of utility programs, such as *diff* and *grep*. You can even buy a LALR(1) parser generator (a freebie under Unix called *yacc*). All of this is in addition to the many features of MS DOS itself that were, uh, borrowed directly from Unix without so much as one word of acknowledgement.

One could spend several thousands of dollars equipping an MS DOS machine with all these programs, and the result would still not approach the power of Unix. A purchase of Xenix seems to me to be the best buy in the software market. It is the most useful piece of bundled software I've seen.

David F. Ziffer  
Software Development Systems  
3110 Woodcreek Dr.  
Downers Grove, IL 60515

# SAS Institute Inc. Announces

## Lattice C Compilers for Your IBM Mainframe

### Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

### One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

### Today...

Our efforts are complete—we have a first-rate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

#### ■ Generation of reentrant object code.

Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.

#### ■ Optimization of the generated code.

We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.

#### ■ Generated code executable in both 24-bit and 31-bit addressing modes.

You can run compiled programs above the 16 megabyte line in MVS/XA.

#### ■ Generated code identical for OS and CMS operating systems.

You can move modules between MVS and CMS without even recompiling.

#### ■ Complete libraries.

We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library

routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix®-style I/O access method.

■ **Built-in functions.** Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix linker and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

### Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

### C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

### Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

### For more information...

Complete and mail the coupon today. Because we've got the development tool for your tomorrow.



SAS Institute Inc.  
SAS Circle, Box 8000  
Cary, NC 27511-8000  
Telephone (919) 467-8000 x 7000

#### I want to learn more about:

- the C compiler for MVS software developers
- the C compiler for CMS software developers
- the cross-compiler with PLINK86 and PLIB86

#### today...so I'll be ready for tomorrow.

Please complete or attach your business card.

Name \_\_\_\_\_

Title \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ ZIP \_\_\_\_\_

Telephone \_\_\_\_\_

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000, Cary, NC, USA.  
27511-8000. Telephone (919) 467-8000, x 7000

## Sixth Generation Minds

*Our man from muppetland returns with an entourage including strange attractors and motorboats on mercury pools.—ed.*

How might we simulate actual human cognition via new kinds of hardware?

A clue comes from the work of Erich Goldmeier, whose essentially gestalt investigations in visual perception were a response to "the frustrating effort to teach pattern recognition" to computers. He found that the distinctions humans make between figure and ground, matter and form, norms and distortion, etc., cannot be the result of examining the features of the figures or any formal aspects of the figures. Instead, Goldmeier believes, there are prototypical or archetypal figures existing somehow at the neurological level, such as "regions of resonance" in the brain.

All this fits with Michael Doherty's latest *DDJ* article citing the works of Rosch, Nelson, and Palermo, which suggests that concepts cannot be broken down into a list of features via reductionistic techniques and then recombinant. In this view, objects

by Richard Grigonis

*Richard Grigonis is employed by Children's Television Workshop. He is best known to *DDJ* readers for his articles on fifth and sixth generation computing and the Grigonis-Doherty debate these articles precipitated.*

are categorized not by tabulating features but in terms of measured distance from some mental ideal or type. Doherty also mentions the linguistic curiosities known as "squishes." The facts that the gerund "his going" can be used in the same positions as a normal noun phrase and that it can be placed along a continuum of "nouniness" implies that one can't assign linguistic terms to discrete categories and state hard rules based on membership in the categories.

But how does all this relate to building a computer whose workings parallel those of the brain?

Until recently, it was thought that the firings of neurons in the brain could be likened to the activity of flip-flops in a computer. Now it appears that large groups of neurons work in unison, interacting via complex electromagnetic fields—which could explain the "regions of resonance" required by Goldmeier's view and also such phenomena as associative memory and the ability of the brain to recognize internally complex stimuli (such as a friend's face) almost instantly.

In one model, which is based on the work of such researchers as E. Roy Johns of the New York University Medical Center and W. Ross Adey of the Loma Linda Veterans Administration Hospital, neurons behave as complex nonlinear oscillators. This places the problem of understanding the collective behavior of neurons in the realm of chaos theory, which describes mathematically systems that shift from periodic to nearly

chaotic behavior—such as the way a stream of air becomes turbulent near an airfoil—that are best described via the mathematical entities known as "strange attractors."

Unlike what happens in more tractable systems, when one changes input to a strange attractor slightly one ends up with a wildly different output. Erol Basar of the University of Physiology in Lübeck, West Germany, plotted the amplitudes of two brain frequencies and found their relationship to be that of a strange attractor. Maybe free will is a strange attractor?

In any case, building a "field-effect" computer based on nonlinear oscillators ought to be a technological nightmare. Eric J. Lerner says that it could be done in one of four ways: (1) microwave circuits built from conventional components; (2) Josephson junctions, which are natural nonlinear oscillators in the microwave region; (3) using an optical processor, increasing the electromagnetic frequencies of individual transmitting units; (4) using some kind of molecular processor such as those projected for future "biochips."

One could think up some additional, perhaps less plausible, versions of such a computer: a swimming pool filled with liquid mercury, the surface of which is disturbed by thousands of toy motorboats with broken propellers and faulty engines (the "non-linear accelerators"); or a soundproofed room filled with thousands of microcomputers fitted with speech recognition/production devices squawking

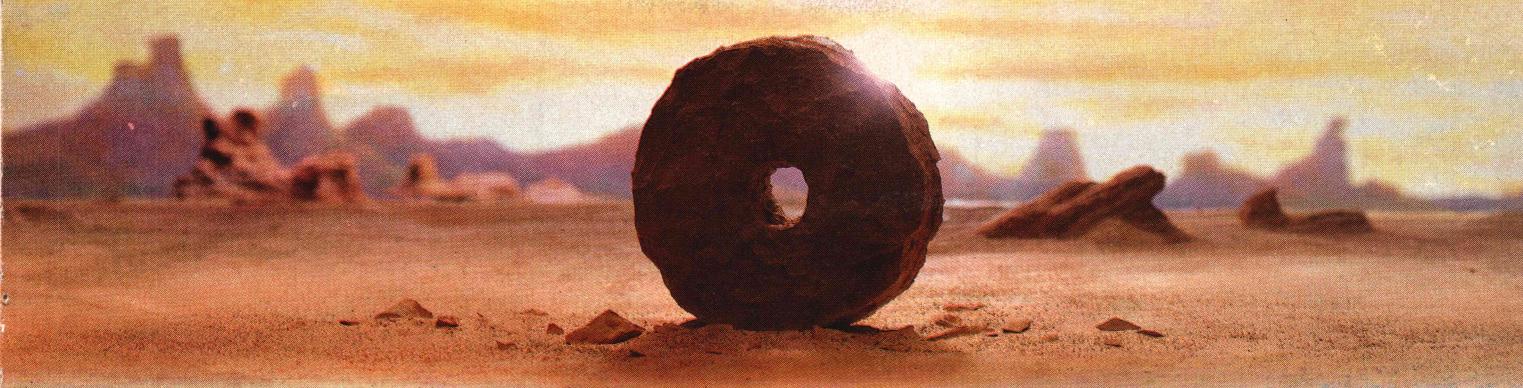
at each other at various frequencies.

But is it worthwhile?

The successes of limited expert systems demonstrate special deficiencies in human cognition. As Doherty wrote, "The strange fact about AI has always been that it's easier to simulate an expert than to simulate the common sense of a five-year-old." The tremendous contextual knowledge that children possess is wired into the brain along with the innate capacity to understand the grammars and semantics of natural language. These abilities have been genetically handed down to us by our ancestors, useful evolutionary "inventions" by the ubiquitous DNA molecule to keep itself from extinction. Thus, the amount of learning required in these cognitive areas is minimal, as they are processes acquired over time in a way that is "transparent to the user."

But we should not feel too complacent. The other side of the coin—learning new forms of knowledge and mastering them as an expert—is a domain easily dominated by computers. If you or I were to memorize a few hundred rules and apply them logically, we could get straight As on MIT calculus finals, but we are unlikely just now to run off to examine Slagle's hundred rules. The future of the art and science of expertise belongs to machines, not to human beings.

DDJ



# SOME HISTORIC BREAKTHROUGHS DON'T TAKE AS MUCH EXPLAINING AS COMPUERVE.

**But then, some historic breakthroughs could only take you from the cave to the tar pits and back again.**

CompuServe, on the other hand, makes a considerably more civilized contribution to life.

It turns the personal computer into something useful.

CompuServe is an information service. Just subscribe, and 24 hours a day, 7 days a week, a universe of information, entertainment and communications is at your service.

## A few of the hundreds of things you can do with CompuServe:

### COMMUNICATE

**Easyplex™** Electronic Mail puts friends, relatives and business associates in constant, convenient touch.

**CB Simulator** lets thousands of enthusiastic subscribers "chatter away" on 72 different channels.

**Over 100 Forums** welcome you to join their online "discussions." They're for everyone from computer owners and gourmet cooks to physicians and game players.

**Bulletin Boards** let you "post" messages where thousands will see them.

### HAVE FUN

**Our full range of games** includes "You Guessed It!," the first online TV-style game show played for real prizes; *Mega-Wars III*, the ultimate in interactive excitement; board; parlor; sports and educational games.

### SHOP

**THE ELECTRONIC MALL™** gives you 'round the clock shopping for name brand goods and services at discount prices from nationally known stores and businesses.

### SAVE ON TRIPS

**TWA Travelshopper™** lets you scan schedules and fares, find the best bargains and order tickets online.

**A to Z Travel/News Service** provides latest travel news plus complete information on over 20,000 hotels worldwide.

### MAKE PHI BETA KAPPA

**Grolier's Academic American Encyclopedia's Electronic Edition** is a complete, constantly updated general reference encyclopedia.

**The College Board**, operated by the College Entrance Examination Board, helps you prepare for the SAT, choose a college and get financial aid.

### BE INFORMED

**The AP News Wire** (covering all 50 states and the nation), the Washington Post, USA TODAY Update and business and trade publications are constantly available. And our electronic clipping service lets us find, clip and file specific news for reading at your convenience.

### INVEST WISELY

**Comprehensive Investment Help** includes complete statistics on over 10,000 NYSE, AMEX and OTC securities. Historic trading statistics on over 50,000 stocks, bonds, funds, issues and options. Five years of daily commodity quotes. Standard & Poor's Value Line. And over a dozen other investment tools.

**Site II** provides demographic and sales potential information by state, county and zip code for the entire country.

### And now for the pleasant surprise.

Although CompuServe makes the most of any computer, it's a remarkable value. You get low start-up costs, low usage charges and local-phone-call access in most major metropolitan areas.

### Here's how to use CompuServe.

CompuServe is "menu-driven," so beginners can simply read the lists of options on their screens and then type in their selections.

Experts can just type in "GO" followed by the abbreviation for whatever topic they're after.

In case of confusion, typing "H" for help brings immediate instructions.

And you can ask general questions either online through our free Feedback service or by phoning our Customer Service Department.

### How to subscribe.

To access CompuServe, you'll need a CompuServe Subscription Kit; a computer, terminal or communicating word processor; a modem and in some cases, easy-to-use communications software.

With your Subscription Kit, you'll receive a \$25 usage credit, a complete hardcover Users Guide, your own exclusive user ID number and preliminary password, and a subscription to CompuServe's monthly magazine, *Online Today*.

Subscription Kits are available in computer stores, electronic equipment outlets, retail stores and catalogs. You can also subscribe with materials you'll find packed right in with many computers and modems sold today.

**Make a move of historic proportions. Subscribe to CompuServe today.**

To receive our free informative brochure or to order direct, call or write:

# CompuServe®

Information Services  
P.O. Box 20212, 5000 Arlington Centre Blvd.  
Columbus, OH 43220

**800-848-8199**

In Ohio, call 614-457-0802

## DDJ Goes On Line

On January 1, the Electronic Edition of *Dr. Dobb's Journal* appeared on CompuServe. Through the Electronic Edition *DDJ* will offer the following:

- We will make available in our data libraries most of the listings that appear in the articles and columns of *Dr. Dobb's Journal* every month. The data libraries will also contain selections from some of our more popular back issues and occasionally articles and listings that have not appeared in the magazine.
- Some of our more significant programs will be available through Softex in

the near future. Softex is CompuServe's electronic software exchange, a menu-driven program that allows users to purchase and download software through the network.

- We will maintain a display area where CompuServe users can read abstracts of the material in the current issue. The display area will also contain general magazine information, such as our editorial calendar, writers' guidelines, and information for advertisers.
- We will compile lists of the commercial software development tools available to microcomputer programmers. We will also compile selected bibliographies and give capsule re-

views of newly released books.

- We will provide a messaging facility with columnists and other SIG members.
- We will stage regular online teleconferences with authors, columnists, and other distinguished guests.
- We will take subscriptions for *Dr. Dobb's Journal*, the magazine. We will also provide a way for readers to register circulation complaints.
- We will provide complete information about other *DDJ* publications, such as back issues, bound volumes, indexes, *Dr. Dobb's Books*, and *Dr. Dobb's Software*. We will also take orders for these items.

Please watch this de-

partment for further announcements about conference schedules or the availability of listings from back issues. Also, if you have a request for a listing from a back issue, drop us a note or just leave a message in the *DDJ* Electronic Edition on CompuServe.

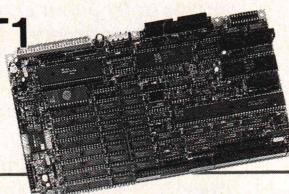
Some of these services may not yet be available. Most will be free, but some will involve a small charge.

You access the *DDJ* Electronic Edition by typing go *DDJ* at any CompuServe system prompt. We hope you will drop by and have a look. See you soon!

DDJ

### MSC-LAT1

\$649



KAYPRO™ users can share the advantage to LAT1. Just take off your main KAYPRO board and put LAT1-K into your cabinet.

All advantage of LAT1 is yours now!

### ZENET NETWORK through twist pair

- 6Mhz HD64B180 (Z80 upward compatible) 512K byte on board (256K installed, 384K RAM/DISK)
- LAN: ZENET port 800K baud CSMA CD twist pair bus type upto 500 meters HDLC
- Floppy: 3.5, 5 and 8 inch, d/s density, d/s sided and d/s track automatic density/format checking
- Hard disk: SCSI interface on board
- Video: 80 X 24 characters (color) and 640 X 200 pixels color graphic 128K byte video RAM character set is downloaded from disk
- Timer: battery back up calendar

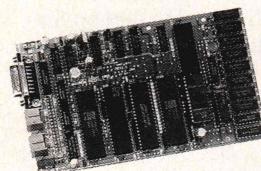
- Serial: RS232C X 2 and TTL X 1
- Parallel: centronics type, 16 bit TTL, 7/8 bit keyboard port (32 characters FIFO)
- O.S.: Turbo Dos, MP/M (multiuser) banked CP/M plus (single user)
- Size: 10 X 6 inch 4 layered
- Assembled and tested
- BIOS source code available
- Completely faster than other Z80SBC

#### MSC-PCX

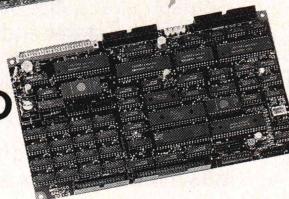
8088 expansion card for LAT1 soon available

### MSC-ICO

\$499



### MSC-MTC



### WORLD SMALLEST COMPUTER

- Full personal CP/M system in palm, 4mhz Z80 256K RAM (128K RAM/DISK)
- Serial: RS232C X 2 automatic baud rate checking
- Parallel: centronics type printer port
- Floppy: 3.5 inch micro floppy disk drive 800K byte (option 5, 3.5 inch drive d/s sided d/s track, automatic density checking)

- O.S.: CP/M plus bank version
- BIOS source code available
- Completely faster than other Z80SBC

#### MSC-MTC/P

Full assembled pcb of MTC  
Under \$189 in OEM quantity

\$299

### Full featured CP/M plus system

- Z80 4mhz 128K Byte RAM Floppy: 3.5, 5 and 8 inch d/s density, d/s sided and d/s track upto 4 disk drives Automatic density/format check
- Serial: RS232C X 2
- Parallel: Centronics type, 16 bits I/O, 7/8 bit keyboard port
- Timer: battery back up calendar
- Video: 80 X 24 high speed CRT controller
- O.S.: CP/M plus bank version included
- Size: 10 X 6 inch 4 layered

- BIOS source code available
- DRI CP/M plus manual \$50
- New word word processor program for MSC-ICO ADD \$50
- Completely faster than other Z80SBC

#### MSC-HCS

Expansion card for ICO  
RAM disk (upto 2M byte) and SCSI hard disk  
interface card for ICO with installation program

\$199

CP/M plus is a registered trademark of Digital Research Inc.  
Z80 is a registered trademark of Zilog Inc.

Turbo Dos is a registered trademark of Software 2000 Inc.  
Mountain Side Computer and ZENET are trademark of Southern Pacific Limited

#### Distributors

England-Quanta systems 01-253-8423  
Denmark-Danbit 03-662020  
Finland-BB Soft 90-692-6297  
India-Betamatix PVT Ltd. 0812-71989  
Australia-LAMRON PTY. Ltd. 02-808-3666

#### Manufacturer and international distributor

#### SOUTHERN PACIFIC LIMITED

Sanwa Bldg., 2-16-20 Minamisawai, Nishi, Yokohama, JAPAN 220  
Phone: 045-314-9514 Telex: 382230 SPACIF J  
Advanced single board computer technology company

#### USA distributor

#### SOUTHERN PACIFIC

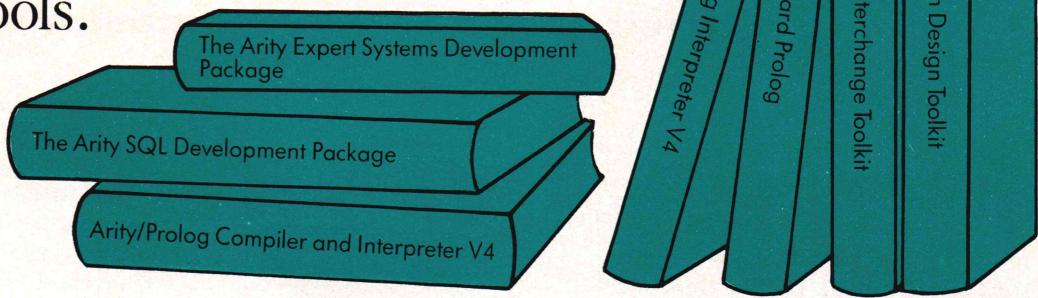
COMPUTER PRODUCTS U.S.A., INC.

21 Altarinda Rd. Orinda, CA 94563

Phone: 415-253-1270

Dealer and distributor inquiries welcome

# Why your next generation of products should use our 5th generation tools.



Arity's integrated family of programming tools allows you to combine software written in Arity/Prolog, the best of the fifth generation languages, with Arity SQL, the best of the fourth generation languages, and with conventional third generation languages such as C or assembly language to build your smarter application.

You can use Arity/Prolog to build expert systems using the Arity Expert Systems Development Package. Or to build natural language frontends. Or to build intelligent information management systems. Arity/Prolog lets you build advanced technology into your vertical applications package.

## And more...

That's not the whole story. Arity's products are all designed to be fast, powerful, serious. Each of our products contains unexpected bonuses. Such as a one gigabyte virtual database integrated into Arity/Prolog. The most powerful of its kind on a PC.

## Quality first. Then price.

In order to be the best, we had to prove it to our customers. Our tradition of quality software design is reflected in every product we sell. Quality first. Then price. And we always provide the best in customer support.

Our products are not copy protected. We do not charge royalties. We offer generous educational and quantity discounts. And we have a 30 day money back guarantee.

Try us to know that we keep our promise on commitment to quality and reliability. Try us by using our electronic bulletin board at 617-369-5622 or call us by telephone—you can reach us at 617-371-2422.

Or fill in this coupon. Whether you order today or not, let us send you full descriptions of our integrated family of Arity products.

**arity**

We design and distribute high quality, serious application software for the IBM PC, XT, AT and all MS-DOS compatibles.

Circle no. 121 on reader service card.

Please complete this form to place your order and/or request detailed information.

Quantity      Info only

Arity Prolog Compiler and Interpreter V4	\$795.00	_____	_____
Arity Prolog Interpreter V4	\$350.00	_____	_____
Arity Standard Prolog	\$ 95.00	_____	_____
Arity SQL Development Package	\$295.00	_____	_____
Arity Expert System Development Package	\$295.00	_____	_____
Arity Screen Design Toolkit	\$ 49.95	_____	_____
Arity File Interchange Toolkit	\$ 49.95	_____	_____

TOTAL AMOUNT (MA residents add 5% sales tax) (These prices include shipping to all U.S. cities)

\$ \_\_\_\_\_

NAME \_\_\_\_\_

SHIPPING ADDRESS \_\_\_\_\_

CITY/STATE/ZIP \_\_\_\_\_

TELEPHONE \_\_\_\_\_

Payment:  Check  PO  AMEX  VISA  MC

Card # \_\_\_\_\_ Exp. date \_\_\_\_\_

Signature \_\_\_\_\_

ARITY CORPORATION • 358 BAKER AVENUE • CONCORD, MA 01742

**arity**

## A New Shell for MS DOS (continued)

This month's column continues with the MS DOS shell. Last month I described how to use the shell and printed Listing One, the shell itself. This month, because the various subroutines that make up the shell are commented well enough so that additional comments here are unnecessary, I'll discuss only the shell's organization at a reasonably high level and those subroutines whose function is not immediately evident. Refer to the listings for more details. This month's listings (Listings Two, Three, Four, and Five, pages 66-74) are the history, shell variables, alias support, and a couple minor support routines. Next month I'll finish up the shell with another collection of miscellany.

A bug was found in last month's listing after the issue went to press. As printed, the shell would try to expand \* or ? even if these characters were in a quoted string. To fix this problem, replace the subroutine

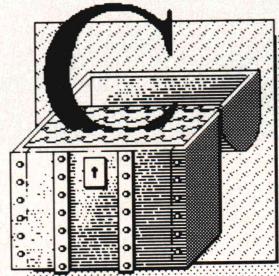
```
has_wild()
```

(line 348) with the code shown in Table 1, page 18.

This morning I dumped all the various shell listings out on my printer for the first time. I admit I'm surprised by the number of pages now piled on my desk (for all that, sh.exe is only 28,906 bytes compared with 23,210 bytes for command.com). Anyway, for both my own sanity and to make everyone's life a little easier, I've compiled a cross-reference of all the shell-related subroutines (Table 2, page 18). The table shows the subroutine name, the listing number (and month of publication), and the line number on which

by Allen Holub

the subroutine starts. Next month's listings are included in the table as are subroutine-like macros.



### Compiling the Shell

The shell was compiled using the most recent version (3.0) of the Microsoft C compiler. (I am not as impressed with the compiler as many other reviewers seem to be; there will be an extensive review of the compiler in this column next month.) I've tried to restrict myself to those library subroutines that are readily portable to other compilers. Most of the system-level routines (`chdir()`, `getcwd()`, and so on) are simple BDOS calls anyway, so it shouldn't be too hard to write them if you don't have them already. All the library routines used are listed in a block of externs on lines 110-123 of Listing One.

Three routines may cause trouble if you're not using the Microsoft compiler. These are `signal()`, which lets you handle a ^C, and the environment manipulation routines `getenv()` and `putenv()`. If you need a `signal()`, I'd suggest using Ray Duncan's *break.asm* (DDJ, September 1985, pp. 119-121), which is functionally very similar to `signal()`. (*Break.asm* sets a global flag rather than calling a subroutine when ^C is encountered.)

`Getenv()`, used to examine an environment string, is relatively straightforward to write. A pointer to the environment string is part of a program's PSP. [For more information about the PSP, see *The Peter Norton Programmer's Guide to the IBM PC* (Microsoft Press, 1985) p. 260f.] In any event, Lattice, Microsoft, and Aztec all have a `getenv()` in their libraries. Unfortunately, of the three, only Microsoft has a `putenv()`, and adding an environment string is a harder prob-

lem, mostly because a child process must inherit the shell's environment. The various fork/spawn functions have to be rewritten to pass the new environment to the child. If anyone has done any of this, please send me your routines, and I'll print them. Lacking that, I'll try to write them myself within the next few months.

The shell modifies only two environment strings. `CMDLINE` holds the full 2,048-byte command line, and `SHLEV` holds the current shell nesting level. If you can live without these, then you don't need a `putenv()`. An alternative approach is to use the Intra-Application Communications Area supported by DOS. The ICA is a 16-byte block at addresses 0000:4f0 to 0000:4ff, reserved by DOS so that programs can communicate with each other (that is, DOS promises not to trash it for you). The current shell level and a long pointer to the command line buffer could be put into the ICA and then accessed by a child process instead of using environment variables. You should probably put a checksum in the ICA as well, to make sure that another program hasn't modified it.

I've used the `void` type in the shell so that a warning will be printed if you use the return value of a subroutine that doesn't return a value, though I haven't used any pointers to `void`. I'd suggest putting a

```
typedef int void;
```

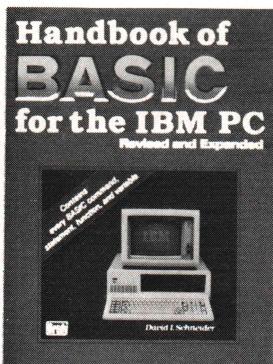
into your stdio.h file rather than dispensing with the `void` declarations—your program will be more portable that way. Microsoft also supports strong type checking, so I've been using it too. Strong type checking is turned on by including a type list as part of an `extern` statement. For example:

```
extern int fopen( char *, char * );
```

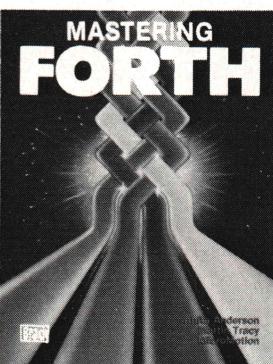
will do all the usual things, and a

# BRADY Speaks Your Language

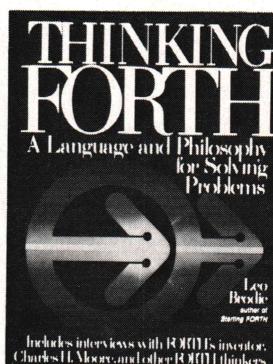
We can help you buff up your BASIC...Put a finish on your FORTH...  
Conquer C...Succeed with Assembler...And more! Just call toll-free  
or use the coupon to order today!



1. PC magazine calls it: "A truly remarkable book...A treasure trove of useful programming information for the IBM PC." \$22.95

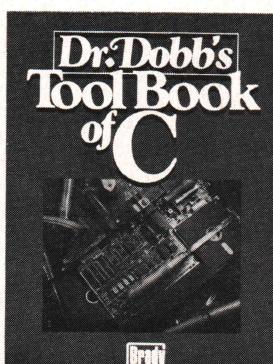


2. A step-by-step tutorial for the high-level, stack-oriented FORTH-83 standard. \$17.95

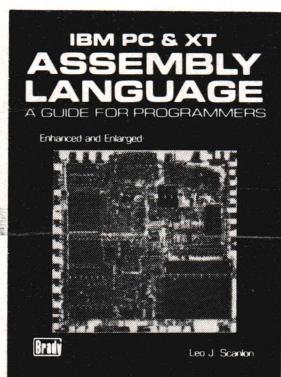


Includes interviews with FORTH's inventor, Charles L. Moore, and other FORTH thinkers.

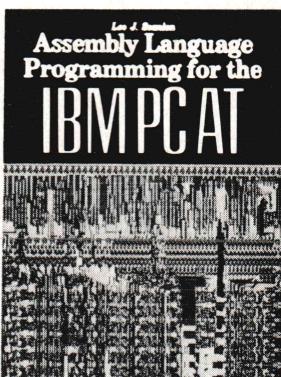
3. The FORTH authority, Leo Brodie, illustrates the elegant logic behind FORTH and shows how to apply specific problem-solving tools to software. \$16.95



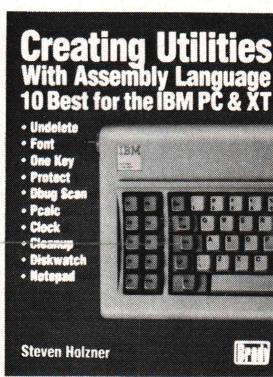
4. The best C articles from the highly respected *Dr. Dobb's Journal* dealing exclusively with C language and programming techniques. \$24.95



5. Our best-selling assembler book has been made even better! It now includes 30 assembler Macros and version 2.0 of the IBM Assembler. \$21.95



6. The author of our best-selling assembler books now demonstrates his detailed and accurate style on the 80286 chip. \$21.95



7. For the more advanced user familiar with Assembly language, here's an opportunity to unleash the power of 10 DOS-enhancing programs. \$21.95



8. Probes the inner workings of the 8086 (used by the AT&T 6300 and 8088 (IBM PC) chips...and describes specific techniques for using the full capability of these chip designs while programming in assembler. \$18.95

Now at your book or computer store.  
Or order toll-free today:

**800-624-0023**

In New Jersey:  
800-624-0024

**Brady** COMMUNICATIONS COMPANY, INC.  
c/o Prentice Hall,  
P.O. Box 512, W. Nyack, NY 10994

Circle the numbers of the titles you want below.  
(Payment must be enclosed; or, use your charge card.) Add \$1.50 for postage and handling.  
Enclosed is check for \$\_\_\_\_\_ or charge to  
 MasterCard  VISA.

1 (0-89303-510-6)  
5 (0-89303-575-0)

2 (0-89303-660-9)  
6 (0-89303-484-3)

3 (0-13-917568-7)  
7 (0-89303-584-X)

4 (0-89303-599-8)  
8 (0-89303-424-X)

Acc't # \_\_\_\_\_ Exp. date \_\_\_\_\_

Signature \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_  
(New Jersey residents, please add applicable sales tax.)  
Dept. 3

G-LDDJ-AL(6)

warning will be printed if the return value from *fopen* isn't put into an *int*. In addition, warnings will be printed if *fopen* is called with anything other than two, character-pointer arguments. A subroutine with a variable number of arguments can be declared:

```
extern void printf( char * , )
```

Here, *printf* must have one character-pointer argument, but this one can be followed by zero or more additional arguments of indeterminate type. *Printf*'s return value (which is garbage anyway) can't be used for anything because of the *void*. If you're not using a compiler that supports strong type checking, don't enter the type list when you type in the code. All the *extern* statements are grouped near the top of every module so that you can find them easily.

Another potential portability problem is the enumerated type (*TOKEN*) used on lines 193–208 of Listing One. You can replace the *TOKEN* declaration with

```
typedef int TOKEN;

#define ALIAS 0
#define CD (ALIAS + 1)
#define CMD (CD + 1)
#define EXIT (CMD + 1)
#define HISTORY (EXIT + 1)
```

One final portability issue: I've used both bit fields and explicit initializers all over the place. To my mind, these are both part of the C language, and I wouldn't consider dispensing with them any more than I'd consider not using *++* or pointers. Just because a few turkey compilers (such as BDS) can't compile a program written in C, I'm not going to deliberately cripple my own programs by not using perfectly legitimate constructs. This is an MS DOS shell anyway, and most MS DOS compilers support bit fields and initializers (if yours doesn't, you should consider getting another compiler). Those that don't support bit fields will just ignore the

```
int has_wild( bp )
register char *bp;
{
    /* Return true if the string has a * or ? in it
     * Does not recognize a * or ? in a quoted string.
     */
    register int inquote = 0;

    for( ; *bp ; ++bp )
    {
        if( *bp == '\' \' && *(bp+1) )
            ++bp;

        else if( ISQUOTE(*bp) )
            inquote = inquote;

        else if( inquote && (*bp == '*' || *bp == '?') )
            return 1;
    }
    return 0;
}
```

**Table 1**

name	listing #	month	module	line number
typedef TOKEN	listing 1	(jan)	sh.c	193
typedef VAR	listing 3	(feb)	var.c	29
add_entry	listing 7	(mar)	dir.c	247
add_hist	listing 2	(feb)	hist.c	72
alias	listing 1	(jan)	sh.c	1237
cd	listing 1	(jan)	sh.c	1313
clab	listing 7	(mar)	dir.c	349
cmds	listing 1	(jan)	sh.c	1417
cmp	listing 7	(mar)	dir.c	359
command_inpu . . .	listing 1	(jan)	sh.c	285
copy_path	listing 7	(mar)	dir.c	327
cptolower	listing 2	(dec)	cptolow.c	1
cpy	listing 3	(dec)	cpy.c	1
cursize	listing 4	(dec)	vidbios.c	49
del_dir	listing 7	(mar)	dir.c	412
DIAG	listing 1	(jan)	sh.c	156
digit	listing 1	(jan)	sh.c	251
dir	listing 7	(mar)	dir.c	431
dirtoa	listing 7	(mar)	dir.c	200
disk_present	listing 1	(jan)	sh.c	1266
doargs	listing 1	(jan)	sh.c	1100
docmd	listing 1	(jan)	sh.c	883
doenv	listing 1	(jan)	sh.c	1359
efgets	listing 5	(dec)	efgets.c	393
eget_hist	listing 2	(feb)	hist.c	169
egets	listing 5	(dec)	efgets.c	138
ENDTRACE	listing 1	(jan)	sh.c	151
errmsgs	listing 1	(jan)	sh.c	1029
execute	listing 1	(jan)	sh.c	631
exp_dir	listing 1	(jan)	sh.c	479
exp_vars	listing 1	(jan)	sh.c	706
file_input	listing 1	(jan)	sh.c	305
find_first	listing 7	(mar)	dir.c	45
find_next	listing 7	(mar)	dir.c	67
findvar	listing 3	(feb)	var.c	43
fixup_name	listing 7	(mar)	dir.c	137

(Continued on page 20)

**Table 2:** Cross-reference for all shell-related subroutines and subroutine-like macros

# PERFORMANCE PACKAGE

Blaise Computing Inc. introduces the PERFORMANCE PACKAGE™ for Turbo Pascal programmers.

## TURBO PASCAL

### Turbo

### ASYNCH™

With Turbo ASYNCH, you can be in constant touch with the world without ever leaving the console. Rapid transit at its best. Turbo ASYNCH is designed to let you incorporate asynchronous communication capabilities into your Turbo Pascal application programs, and it will drive any asynchronous device via the RS232 ports, like printers, plotters, modems or even other computers. Turbo ASYNCH is fast, accurate and lives up to its specs. Features include...

- ◆ Initialization of the COM ports allowing you to set all transmission options.
- ◆ Interrupt processing.
- ◆ Data transfer between circular queues and communications ports.
- ◆ Simultaneous buffered input and output to both COM ports.
- ◆ Transmission speeds up to 9600 Baud.
- ◆ Input and output queues as large as you wish.
- ◆ XON/XOFF protocol.

The underlying functions of Turbo ASYNCH are carefully crafted in assembler for efficiency, and drive the UART and programmable interrupt controller chips directly. These functions, installed as a runtime resident system, require just 3.2K bytes. The interface to the assembler routines is written in Turbo Pascal.

The Turbo Pascal PERFORMANCE PACKAGE™ is for the serious Turbo Pascal programmer who wants quality tools to develop applications. Every system comes with a comprehensive User Reference Manual, all source code and useful sample programs. They require an IBM PC or compatible, utilizing MS-DOS version 2.0 or later. There are no royalties for incorporating PERFORMANCE PACKAGE functions into your applications.

Turbo POWER TOOLS and Turbo ASYNCH sell for \$99.95 each, and they may be ordered directly from Blaise Computing Inc. To order, call (415) 540-5441.

BLAISE COMPUTING INC.

# BLAISE

# Watch us!

- ◆ 2034 BLAKE STREET
- ◆ BERKELEY, CA 94704
- ◆ (415) 540-5441

**Turbo POWER TOOLS™** Turbo POWER TOOLS is a sleek new series of procedures designed specifically to complement Turbo Pascal on IBM and compatible computers. Every component in Turbo POWER TOOLS is precision engineered to give you fluid and responsive handling, with all the options you need packed into its clean lines. High performance and full instrumentation, including...

- ◆ Extensive string handling to complement the powerful Turbo Pascal functions.
- ◆ Screen support and window management, giving you fast direct access to the screen without using BIOS calls.
- ◆ Access to BIOS and DOS services, including DOS 3.0 and the IBM AT.
- ◆ Full program control by allowing you to execute any other program from within your Turbo Pascal application.
- ◆ Interrupt service routines written entirely in Turbo Pascal. Assembly code is not required even to service hardware interrupts like the keyboard or clock.

Using Turbo POWER TOOLS, you can now "filter" the keyboard or even DOS, and create your own "sidekickable" applications.

YES, send me the Best for the Best! Enclosed is \_\_\_\_\_ for  
 Turbo ASYNCH  Turbo POWER TOOLS. (CA residents add  
6 1/2% Sales Tax. All orders add \$6.00 for shipping.)

Phone: \_\_\_\_\_

Name: \_\_\_\_\_

Shipping Address: \_\_\_\_\_

State: \_\_\_\_\_ Zip: \_\_\_\_\_

City: \_\_\_\_\_

Exp. Date: \_\_\_\_\_

VISA or MC #: \_\_\_\_\_

Turbo Pascal is a trademark of Borland International. Turbo POWER TOOLS, Turbo ASYNCH and PERFORMANCE PACKAGE are trademarks of Blaise Computing Inc. IBM is a registered trademark of International Business Machines Corporation. MS-DOS is a trademark of Microsoft Corporation.

: <num> part of the declarations, so your structures will be a little larger, but the program will still compile without modification.

The subroutines are organized functionally rather than alphabetically. You can use the cross-reference if you need to find something. The biggest potential organizational problem is *sh.c* (Listing One), which is too big for me to be totally comfortable with it. Break it up if you need to. The routines on lines 1137–1379 (the various system-level command support routines: *setenv*, *set*, and so on) could be made into another independent module without too much trouble, but that only gets rid of a few hundred lines.

### Shell Organization

The shell itself (Listing One) is organized functionally into several parts:

*Typedefs*, *#defines*, and so on (lines 110–2226)

Input routines (lines 227–346)

Command processing (lines 347–1026)

Start-up routines (lines 1029–1136)

Internally implemented commands (lines 1138–1378)

The command processor itself (including *main*, lines 1382–1545)

Of these, the organization of the input routines needs some comment. I eventually intend to augment *Sh* with several of the loop control functions supported in the Unix C shell. The easiest way to do this is to treat the shell as a small compiler (or interpreter). That is, it's nice to have a token recognizer parse keywords from the input and then tell the command interpreter what to do rather than have the command interpreter itself figuring out what's on the input line. So, the command interpreter (*cmds()* on line 1417) calls *next\_cmd()* to get a command from input. *cmds()* then strips the first word from the input line and calls *tokenize()* to analyze this word. *Tokenize()* returns a unique integral value that can be used to vector into a switch.

Input can come from one of three places, depending on the command line with which the shell was in-

gcur	listing 4	(dec)	vidbios.c	82
get_hist	listing 2	(feb)	hist.c	97
get_hnum	listing 2	(feb)	hist.c	188
getcur	listing 4	(dec)	vidbios.c	105
getkey	listing 5	(dec)	efgets.c	69
getl	listing 5	(dec)	efgets.c	372
getpage	listing 4	(dec)	vidbios.c	36
getvar	listing 3	(feb)	var.c	194
has_only	listing 7	(mar)	dir.c	113
has_wild	listing 1	(jan)	sh.c	348
haswild	listing 7	(mar)	dir.c	84
hist_name	listing 2	(feb)	hist.c	214
history	listing 2	(feb)	hist.c	269
interactive...	listing 1	(jan)	sh.c	269
isalias	listing 3	(feb)	var.c	25
isname	listing 3	(feb)	var.c	24
isrootdir	listing 7	(mar)	dir.c	97
ISVAR	listing 1	(jan)	sh.c	179
iswhite	listing 1	(dec)	next.c	5
ISWHITE	listing 1	(jan)	sh.c	177
main	listing 1	(jan)	sh.c	1489
mk_dir	listing 7	(mar)	dir.c	385
next	listing 1	(dec)	next.c	8
next_cmd	listing 1	(jan)	sh.c	965
nextarg	listing 1	(jan)	sh.c	441
nextarg	listing 8	(mar)	reargv.c	12
PMODE	listing 1	(jan)	sh.c	186
print_hist	listing 2	(feb)	hist.c	195
printalias	listing 3	(feb)	var.c	173
printvars	listing 3	(feb)	var.c	183
prompt	listing 1	(jan)	sh.c	858
PSTR	listing 1	(jan)	sh.c	160
ptail	listing 5	(dec)	efgets.c	112
pwd	listing 1	(jan)	sh.c	1251
rcopy	listing 1	(jan)	sh.c	915
reargv	listing 8	(mar)	reargv.c	46
reset_fileinput	listing 1	(jan)	sh.c	238
restore_hist	listing 2	(feb)	hist.c	253
save_hist	listing 2	(feb)	hist.c	231
scur	listing 4	(dec)	vidbios.c	68
search	listing 1	(jan)	sh.c	573
set	listing 1	(jan)	sh.c	1172
setargs	listing 1	(jan)	sh.c	1063
setcur	listing 4	(dec)	vidbios.c	100
setenv	listing 1	(jan)	sh.c	1138
setvar	listing 3	(feb)	var.c	137
shift	listing 1	(jan)	sh.c	1344
skipto	listing 6	(mar)	skipto.c	1
SKIPWHITE	listing 1	(jan)	sh.c	178
ssort	listing 9	(mar)	ssort.c	9
strip	listing 1	(jan)	sh.c	363
strsave	listing 5	(feb)	strsave.c	6
tokenize	listing 1	(jan)	sh.c	1393
TRACE	listing 1	(jan)	sh.c	150
unalias	listing 1	(jan)	sh.c	1227
unargv	listing 4	(feb)	unargv.c	7
unsetvar	listing 3	(feb)	var.c	62
usage	listing 1	(jan)	sh.c	1047
use_exit	listing 1	(jan)	sh.c	1382
varcpy	listing 3	(feb)	var.c	101
wchar	listing 4	(dec)	vidbios.c	112
wstr	listing 4	(dec)	vidbios.c	125

**Table 2 (cont.):** Cross-reference for all shell-related subroutines and subroutine-like macros

# Turbo, who?

**Do you have to give up power and advanced potential to get ease of use and affordability? Not anymore. Because now, you can have UCSD Pascal for only \$79.95!**

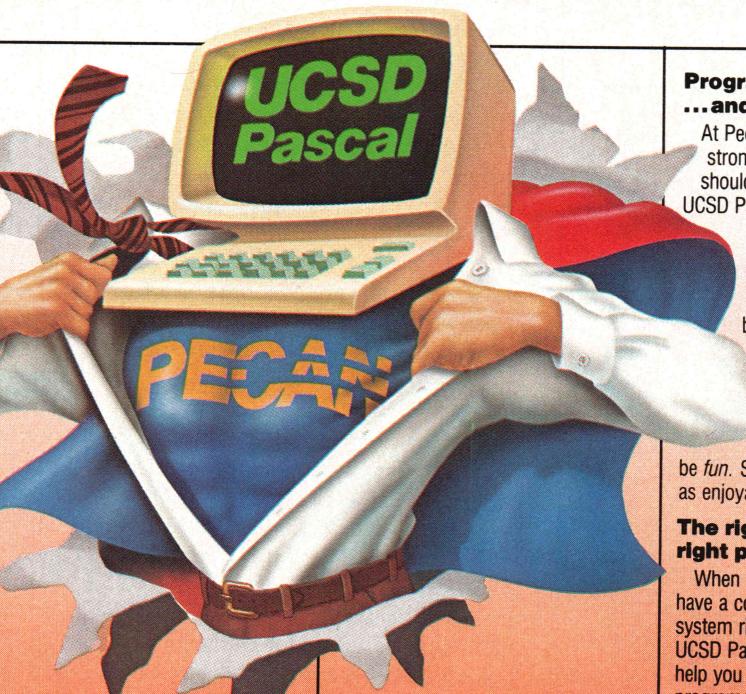
If you're making your move into programming, there's no better way to go than Pascal. And starting now, you *don't* have to settle for a stripped-down version of Pascal in order to get a price that's right. Instead, you can choose UCSD Pascal—the recognized Pascal programming standard in colleges and universities throughout the country—at the incredibly low introductory price of \$79.95 for your PC-DOS, MS-DOS, or other popular computer.

#### Start with the standard

With an *entry-level* system, you spend a lot of valuable time learning a *non-standard* form of Pascal. And you don't get all the capabilities a true Pascal system is supposed to deliver—unless you buy a lot of add-on utilities—which can send the cost of your system sky-high! Worst of all, when you're ready to tackle anything more than short, simple programs—you have no choice but to move up to a more sophisticated system (like UCSD Pascal). And at that point, you also have to *relearn* standard Pascal.

#### UCSD Pascal has everything you need

With UCSD Pascal, you get a



full-featured, professional programming tool that's being used right now in the development of major scientific and business applications. The system comes with an outstanding text editor, a complete on-line tutorial, 8087 math coprocessor support and BCD (decimal arithmetic) *included* in the package at no extra cost. In fact, UCSD Pascal contains virtually everything you need—as standard equipment—for developing the simplest to the most

sophisticated programs.

UCSD Pascal is available for MS-DOS, PC-DOS, UNIX, VMS, MSX and many other operating systems. You can use UCSD Pascal to write programs of any size on virtually any computer, and port them to any other computer. And if speed is what you're after, the latest native code version of UCSD Pascal actually benchmarks favorably with Turbo Pascal® in execution time!

#### Programming that's easy...and fun!

At Pecan Software Systems, we strongly believe programming should be as easy as possible. UCSD Pascal was originally designed for teaching programming skills, so it's extremely easy to learn and to use. With UCSD Pascal, you'll be developing programs right from the start that are easy to write, easy to understand, and easy to maintain. We also believe that programming should be *fun*. So we've made UCSD Pascal as enjoyable to use as it is powerful.

#### The right tool at the right price

When the fun gets serious, you'll have a comprehensive programming system right at your fingertips with UCSD Pascal—a system that will help you develop those big-league programs you may eventually want to write—at a price you can readily afford.

Put UCSD Pascal programming power on your PC now for only \$79.95! Order by mail today or phone now 1-800-63-PECAN. UCSD Pascal—the original standard of Pascal programming excellence. The new leader in Pascal price/performance.

# PECAN™

The UCSD Pascal Company  
Pecan Software Systems, Inc.  
1410 39th Street, Brooklyn, NY 11218  
718-851-3100

# UCSD Pascal™

UCSD Pascal \$79.95 (for PC-DOS, MS-DOS, AMIGA, APPLE, ATARI 520, MACINTOSH, RAINBOW, TANDY, as well as most popular 8/16/32-bit systems).  
Price includes 8087 support and BCD.

Please send me \_\_\_\_\_ copies of UCSD Pascal for my \_\_\_\_\_ (Name and model of computer)  
My disk size is 3 1/2" \_\_\_\_\_ 5 1/4" \_\_\_\_\_ 8" \_\_\_\_\_.  
Total amount (NYS add appropriate tax) \_\_\_\_\_.  
Payment by  VISA  MC  US Bank Check  Bank Draft  
Card Number: \_\_\_\_\_ Credit Card Expiration Date: \_\_\_\_\_

UCSD Pascal  
Not copy protected  
60-day money-back guarantee  
CREDIT CARD ORDERS  
CALL TOLL-FREE  
1-800-63-PECAN  
(NYS) 1-800-45-PECAN

Name \_\_\_\_\_  
Shipping Address \_\_\_\_\_  
City \_\_\_\_\_  
Telephone \_\_\_\_\_  
State \_\_\_\_\_  
Zip \_\_\_\_\_

Call or write for UNIX, VAX or other UCSD Pascal versions—and ask about our powerful Pascal add-ons, too. SCHOOLS: Contact us for our special educational discounts!  
Call toll-free or enclose a check with this coupon to place an order. Please add \$2.50 for shipping within the U.S. Foreign orders add \$10 and make payment by bank draft payable in US dollars on US bank. New York State residents add appropriate sales tax.

UCSD Pascal is a registered trademark of The Regents of University of California

# Work Smart with These Powerful C Utilities

Get more value from your C system. Boost program quality and slash development time with these professional utilities for leading C-compiler systems.

## C Utility Library \$185 \$155

Over 300 C subroutines

C and assembler source code and demonstration programs for screen handling, color printing, graphics, DOS disk and file functions, memory management and peripherals control.

## C-tree \$395 \$329

### B-Tree database system

Store, update and retrieve records easily. High-level multi-key ISAM routines and low-level B-Tree functions. Available for MS-DOS, CP/M-86, and CP/M-80. Easily transported. Adaptable for network and multiuser. Includes source.

## PHACT \$295 \$200

### Data Base Record Manager

Includes high-level features found in larger database systems. Available for MS-DOS, CP/M-86 and CP/M-80.

## Pre-C \$395 \$329

### LINT-like source code analyzer

Locates structural and usage errors. Cross-checks multiple files for bad parameter declarations and other interface errors.

## Windows for C \$195 \$165

### Versatile window utility

Supports IBM PC compatible and some non-compatible environments.

## PANEL \$295 \$235

### Screen generating utility

Create custom screens via simple, powerful editing commands. Select colors, sizes and types, edit fields. Includes direct input utility.

## HALO \$280 \$199

### Ultimate C graphics

A comprehensive package of graphics subroutines for C. Supports multiple graphics cards.

## PLINK-86 \$395 \$315

### Overlay linker

Includes linkage editor, overlay management, a library manager and memory mapping. Works with Microsoft and Intel object format.

To order or for information call:

**TECWARE**  
1 800 TEC-WARE

(In NJ call 201-530-6307)  
Circle no. 109 on reader service card.

UNIX is a registered TM of Bell Laboratories. C-tree, TM Farcom, Inc. PHACT TM PHACT ASSOC. Pre-C, PLINK-86, TM PHONIX. HALO TM Media Cybernetics, Inc. PC Int TM GIMPLE Software. PANEL, TM Roundell Computer Systems, Inc. WINDOWS FOR C TM Creator Solutions. CP/M TM DR

## C CHEST

(Continued from page 20)

voked: the command line itself, a file, or interactively from standard input. All three of these sources have their own problems, and I didn't want *next\_cmd()* to have to worry about these problems. Thus, the shell uses multiple input routines.

When *argv* is parsed (by *doargs()*, line 1100), a global pointer to a subroutine (*lfunc*, line 234) is initialized to point at an appropriate input routine. Text is then input indirectly through this pointer. *lfunc* will point at one of the routines *interactive\_input()*, *command\_input()*, or *file\_input()*. All three routines act in the same way—they get a line of input from somewhere and put that line into the global array *lbuf* (declared on line 216). A pointer to *lbuf* is returned on success, and 0 is returned on EOF. The string input routine *command\_input()*, returns EOF when it reaches the end of a string to maintain compatibility with the other routines.

Of the three input routines, the weirdest is *file\_input()* (used to process batch files). The problem here is caused by collusion between MS DOS and the compiler's I/O library. When a program spawns a child process, the child inherits the parent's file descriptors. This is documented in several places, both in the MS DOS and the Microsoft C Compiler documentation. None of these sources, however, deign to mention that when the child process terminates, *exit()* will close all open files, including those files that belong to the parent process. In other words, spawning a child process under MS DOS closes all open files in the parent process as an undesirable side effect. This problem is circumvented in *file\_input()*, which reads a line from a file, remembers the current position in the file with an *fseek()* call, and then closes the file. The next time *file\_input()* is called, it reopens the file, seeks to the previous position in the file, and then reads another line. I know this is a kludge, but I couldn't think of any easy way around the problem short of reading the entire batch file into a local buffer and then processing that buffer. The repeated seeks seemed a better solution, but

it's not a good one.

Another somewhat convoluted piece of code is the routine that does alias expansion. The expansion of a single alias is straightforward. The same routines (and tables) are used to hold and expand both aliases and shell variables. An alias has the high bit of the first character of the name set when it is stored and a shell variable does not so that the alias expansion routines can differentiate between them. That is, aliases and shell variables do not share the same name pool, even though they're stored in the same table. All the simple alias maintenance routines are in Listing Three.

The shell, on the other hand, uses the simple alias expansion routines in complex ways. The problem here is compound commands—single aliases consisting of several, semicolon-delimited commands concatenated together. These additional commands could also be aliases.

The initial command is read in by *next\_cmd()* on line 991. Blank lines and comment lines (those with a # in the far left column) are skipped at this level. Then history is applied. (Note that applying history here means that the ! that signifies a history expansion has to be the character at the left end of the line not the first character of the command, which could be anywhere on the line following a semicolon.) Now aliases are expanded by calling the recursive routine *rcopy()* (on line 915). Each recursive iteration expands one alias. Looking at the code as I write this, the method used seems needlessly convoluted. On the other hand, it works—"If it ain't broke, don't fix it."

## Availability

This column is part of a four-part series describing the entire shell. A reprint of all four parts along with a disk containing the listings and an executable version of the shell is available for \$29.95 from *Dr. Dobb's Journal*, 2464 Embarcadero Way, Palo Alto, CA 94303. Please direct inquiries to the The Shell. Prepayment is required.

**(Listings begin on page 66)**

## Reader Ballot

Vote for your favorite feature/article.  
Circle Reader Service No. 1.

# NEW RELEASE

## Ecosoft's Eco-C88 Rel. 3.0 C Compiler



**\$5995**

Release 3.0 has new features at an unbelievably low price. ECO-C88 now has:

- Prototyping (the new type-checking enhancement)
- enum and void data types
- structure passing and assignment
- All operators and data types (except bit fields)
- A standard library with more than 200 functions (many of which are System V compatible for greater code portability)
- cc and mini-make that all but automates the compile process
- 8087 support (we sense the 8087 at runtime - no dual libraries)
- ASM or OBJ output for use with MSDOS linker
- Tiered error messages - enable-disable lint-like error checking
- Fast compiles and executing code
- Expanded user's manual
- Enhanced CED program editor (limited time offer)

We also offer the following support products for Eco-C88.

### CED Program Editor

CED now supports on-line function help.

If you've forgotten how to use a standard library function, just type in the name of the function and CED gives you a brief summary, including function arguments. CED is a full screen editor with auto-flagging of source code errors, multiple windows, macros, and is fully configurable to suit your needs. You can edit, compile, link, and execute DOS commands from within the editor. Perfect for use with Eco-C88. For IBM PC, AT and look alikes.

**\$2995**



### C Programming Guide **\$20**

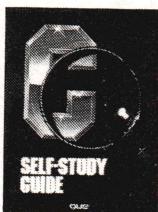
After reading the 1st edition, Jerry Pournelle (BYTE Magazine) said: "I recommend this book... Read it before trying to tackle Kernighan and Ritchie." The second edition expands this best seller and walks you through the C language in an easy-to-understand manner. Many of the error messages include references to this book making it a perfect companion to Eco-C88 for those just starting out with C.

### C Source for Standard Library

Contains all of the source code for the library functions that are distributed with Eco-C88, excluding the transcendental and functions written in assembler.

**\$10**

(\$20 if not with order)



### C Self-Study Guide **\$17**

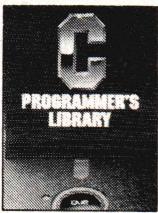
(Purdum, Que Corp.). Designed for those learning C on their own. The book is filled with questions-answers designed to illustrate many of the tips, traps, and techniques of the C language. Although written to complement the Guide, it may be used with any introductory text on C.

### Developer's Library

Contains the source code for all library functions, including the transcendental and those written in assembler. Perfect for the developer that wish to write their own custom functions or learn how we implemented the Eco-C88 library.

**\$25**

(\$50 if not with order)



### C Programmer's Library **\$20**

(Purdum, Leslie, Stegemoller, Que Corp.). This best seller is an intermediate text designed to teach you how to write library functions in a generalized fashion. The book covers many advanced C topics and contains many useful additions to your library including a complete ISAM file handler.

Eco-C88 C compiler requires an IBM PC, XT, or AT (or compatible) with 256K of memory, 2 disk drives and MSDOS 2.1 or later. Call today:

**1-800-952-0472** (for orders)  
or  
**1-317-255-6476** (tech. info.)



**Ecosoft, Inc.**

6413 N. College Ave. • Indianapolis, IN 46220



TRADEMARKS: ECO-C88, ECOSOFT

Circle no. 89 on reader service card.

# THE PROGRAMMER'S SHOP™

helps save time, money and cut frustrations. Compare, evaluate, and find products.

## SERVICES

- Programmer's Referral List
- Dealer's Inquire
- Compare Products
- Newsletter
- Help find a Publisher
- Rush Order
- Evaluation Literature FREE
- Over 700 products
- BULLETIN BOARD - 7PM to 7AM 617-826-4086

## AI - Expert System Dev't

ExpertEASE - Develop by describing examples of how you decide. Call EXSYS - All RAM, Probability. Why. Trees, Solid, files, popular PCDOS \$359 1st Class - by example, interfaces \$250 INSIGHT 1 - Probabilities, required thresholds, menus, fast PCDOS \$ 79 INSIGHT 2 - adds backward, forward, partitions, dB2, lang. access. PCDOS \$399 Others: APES (\$359), Advisor (\$949), ES Construction (\$100), ESP (\$845), Expereteach \$399. Expert Choice (\$449), more.

## AI-LISP

List Our GC LISP - "Common", rich. Interpreter - Interactive Tutorial \$495 Call LARGE Model - 2 to 15 meg. \$695 \$649 Compiler and LM. Interp. \$1190 1045 ExperLisp - Interpreter: Common LISP syntax, lexical scoping, toolbox, graphics. Native Code COMPILER.512K MAC \$465 TLC LISP - "LISP-Machine" - like, all RAM, classes, turtle graphics, 8087, compiler. CPM-86, MSDOS \$225 TransLISP - formerly LISP-86 \$ 75 WALTZ LISP - "FRANZ LISP" - like, big nums, debug, CPM-80 MSDOS \$149 Others: IQ LISP (\$155), BYSO (\$125), MuLISP-86 (\$199)

## AI-PROLOG

ARITY PROLOG - full, debug, ASM, C, virtual. Compiler \$795 MSDOS \$350 MicroProlog - enhanced \$229 MPROLOG - Rich syntax, editor, segment work space, portable. PCDOS \$725 Professional MicroProlog MSDOS \$359 TransPROLOG - Learn Fast. Standard, tutorials, samples MSDOS Call Others: Prolog-1 (\$359), Prolog-2 (\$1895),

## AI-OTHER

METHODS - SMALLTALK has objects, windows, browser, inspector. PCDOS \$239 QNIAL - Combines APL with LISP Library of sample programs included. Source or binary. PCDOS \$375 SNOBOL4 + -great for strings, patterns: MSDOS \$ 85

## FEATURE

TransLisp - "Common subset, tutorial, editor, PP, trace. Best to learn. All MSDOS Only \$ 75

## Free Literature - Compare Products

Evaluate products. Compare competitors. Learn about new alternatives. One free call brings information on just about any programming need. Ask for any "Packet" or Addon Packet  AI  ADA. Modula  BASIC  C  COBOL  Editors  FORTH  FORTRAN  PASCAL  UNIX/PC or  Debuggers, Linkers

## RECENT DISCOVERY

Visual Computer: 8088 - Simulates demos or any .exe, Com. Debugger. 350 pg. tutorial \$ 59

## BASIC

ACTIVE TRACE, DEBUGGER - BASICA, MBASIC, interactive, well liked MSDOS \$ 79 BASCOM-86 - Microsoft 80/86 \$279 BASIC DEVELOPMENT SYSTEM - (BDS) for BASICA; Adds Renum. crossref, compress. PCDOS \$115 BetterBASIC all RAM, modules. structure. BASICA - like PCDOS \$169 8087 Math Support \$ 89 Run-time module \$459 CADSAM FILE SYSTEM - full ISAM in MBASIC source. MSDOS \$ 75 CB-86 - DRI CPM86, MSDOS \$419 Data Manager - full source MSDOS \$325 InfoREPORTER - multiple PCDOS \$115 Prof. Basic - Interactive; debug PCDOS \$ 89 8087 Math Support \$ 47 QuickBASIC by Microsoft - Compiles full syntax of IBM BASICA, 640K, PCDOS \$ 85 TRUE Basic - ANSI PCDOS \$109 Run-time Module \$459

Ask about ISAM, other addons for BASIC

## COBOL

Macintosh COBOL - Full MAC \$459 MBP - Lev II, native MSDOS \$885 MicroFocus Prof. - full PCDOS Call Microsoft Version II - upgraded. Full Lev. II, native, screens. MSDOS \$500 Realia - very fast MSDOS Call Ryan McFarland - portable MSDOS \$695

## Editors for Programming

BRIEF Programmer's Editor - undo, windows, reconfigure PCDOS Call C Screen with source 80/86 \$ 75 EMACS by UniPress - powerful, multifile, windows, DOS, MLISP, programming. Source: \$949 \$299 Entry Systems for C PCDOS \$325 Epsilon - like EMACS, PCDOS \$169 Kedit - like XEDIT PCDOS \$115 FirsTime by Spruce - Improve productivity. Syntax directed for Turbo (\$69), Pascal (\$229), or C (\$239) PMATE - power, multitask 80/86 \$159 VEDIT - well liked, macros, buffers, CPM-80-86. MSDOS PCDOS \$119 XTC - multitasking PCDOS \$ 95

## Ask about Atari ST, Amiga

## C Language - Compilers

BDS C - solid value, fast CPM80 \$125 C86 by CI - 8087, reliable MSDOS Call Consulair Mac C w/toolkit MAC \$299 ECO C/88 MSDOS \$ 59 Lattice C - from Lifeboat MSDOS \$289 Lattice C - from Lattice MSDOS \$339 Mark Williams - debugger MSDOS \$379 Megamax - tight, full MAC \$239 Microsoft C 3.0 - new, MSDOS \$259 Q/C 88 by Code Works - Compiler source, decent code, cross/native MSDOS \$295 Wizard C - Lattice C compatible, full sys. III, lint, fast. MSDOS \$379

## C Language - Interpreters

C-terp by Gimpel - full K & R, .OBJ and ASM, large progs. MSDOS \$249 INSTANT C - Source debug, Edit to Run-3 seconds MSDOS \$399 Introducing C - Interactive C to learn fast, tutorial PCDOS \$115 Professional Run/C has Run/C plus ability to create add-in libraries. (Lattice C compatible) and load/unload them. MSDOS \$199 Run/C - improved MSDOS \$109

## C Libraries - General

Blaise C Tools 1 (\$109), C Tools 2 \$ 89 C Food by Lattice - ask for source \$119 C\*LIB by Vance \$129 C Utilities by Essential - Comprehensive screen graphics, strings, file handling, memory mgmt. Source. MSDOS \$139 Entelekon C Function Library \$119 Entelekon C Windows \$119 Entelekon Superfonts for C \$ 45 Greenleaf Functions - portable, ASM \$149 Polytron - for Lattice, ASM source \$ 99 Software Horizons - Pack 1 \$129

## C Libraries - Communications

Asynch by Blaise \$149 Greenleaf - full, fast \$149 Software Horizons - pack 3 \$119

## C Libraries - Files

FILES: C Index by Trio - full B + Tree, vary length field, multi compiler /File is object only \$ 89 /Pro is partial source \$179 /Plus is full source \$349 CBTREE - multiuser record locking, sequential, source, no royalties \$99

# THE PROGRAMMER'S SHOP™

provides complete information, advice, guarantees and every product for Microcomputer Programming.

We support MSDOS (not just compatibles)  
PCDOS, Xenix-86, CPM-80, Macintosh,  
Atari ST, and Amiga.

## C Libraries - Files Cont

dbVISTA - full indexing, plus optional  
record types, pointers, Network.  
Object only - MS C, LAT, C86 \$179  
Source - Single user \$459  
Source - Multiuser \$929

## C Support - Systems

Basic C Library by C Source \$139  
C Debug - Source debuggers - by  
Complete Soft (\$269), MSD (\$149).  
C Sharp - well supported, Source,  
realtime, tasks MSDOS \$600  
C ToolSet - DIFF, xref, source \$135  
Lattice Text Utilities \$105  
The HAMMER by OES Systems  
H.E.L.P. By Everest Solutions \$179  
\$329

## C - Screens, Windows, Graphics

Curses by Lattice PCDOS \$110  
CView - input, validate PCDOS \$195  
C Power Windows PCDOS \$130  
Databurst - C or Basic PCDOS \$215  
GraphiC - source in C MSDOS \$219  
Topview Toolbasket by Lattice \$219  
View Manager for C by Blaise \$219  
Windows for C - fast \$149  
Windows for Data - validation \$219

## DEBUGGERS

Advanced Trace-86 by Morgan  
Modify code on fly.  
CODESMITH - visual, interactive  
debugger. Symbolize, modify  
and rewrite Assembler  
Periscope I Debugger by Data Base  
Decisions - own 16K  
Periscope II Debugger - load after  
"bombs", symbolic, "Reset Box,"  
2 Screen PCDOS \$119  
Pfix-86 Plus Symbolic Debugger by  
Phoenix - windows \$289  
SOURCE PROBE by Atron for  
Lattice, MS C, Pascal. Windows  
single step, 2 screen, log file.

## Feature

Panel Screen Generator - Create  
screen with editor, generates  
code. Full data validation,  
windows, no royalties. Specify  
Lattice, MSC, C86, MS Fortran  
or PASCAL  
\$239

## SERVICE: FREE NEWSLETTER

Software development and AI on micros: trends, forecasts, controversies, innovations, and techniques. Plus an announcement of 80 NEW tools. CALL for the "Newsletter Packet."

## Fortran & Supporting

Forlib + by Alpha - graph, comm. \$ 59  
Fortran > C - FORTRIX creates  
maintainable translations. MSDOS \$995  
MACFortran by Microsoft - full '77 \$239  
MS Fortran \$239  
No Limit - Fortran Scientific \$129  
PolyFortran - xref, pp, screen \$149  
Prospero - '66, reentrant \$390  
RM Fortran - enhanced "IBM Ftn" \$429  
Scientific Subroutines - Matrix \$149  
Statistician by Alpha \$269  
Strings and Things - registers, shell \$ 59

## MultiLanguage Support

BTRIEVE ISAM MSDOS \$199  
CODESIFTER - Execution PRO-  
FILER. Spot bottlenecks. Symbolic.  
automatic. PCDOS \$109  
MultiHALO Graphics-Multiple video  
boards, printer, rich. Animation,  
engineering, business.  
ANY MS language, Lattice, C86 \$189  
For Turbo \$ 89

PLINK 86 - a program-independent  
overlay linker to 32 levels for all MS  
languages, C86 and Lattice. \$299  
PFinish Performance Analyzer  
by Phoenix \$219  
Profiler by DWB Associates \$109  
PS MAKE by UniPress \$ 79  
Screen Sculptor - slick, thorough,  
fast, BASIC, PASCAL. PCDOS \$109  
ZAP Communications - VT 100, TEK  
4010 emulation, full xfer. PCDOS \$ 85

## TURBO PASCAL and SUPPORT

BORLAND: Turbo 3.0 \$ 49  
3.0 with 8087 or BCD \$ 79  
3.0 with 8087 and BCD \$ 85  
Turbo Graphix - graphs, windows \$ 39  
Turbo Toolbox or Editor \$ 55  
Turbo Tutor \$ 29  
TURBO... Asynch by Blaise, full \$ 89  
MetaWindow by Metagraphics \$ 49  
Power Tools by Blaise - library \$ 89  
Power Utilities - profiler, pp \$ 89  
Professional - interrupts, macros, \$ 50  
OTHERS: Screen Sculptor (\$99),  
Pascal Pac (\$100), Tidy (\$45),  
Multi Halo (\$89).

## RECENT DISCOVERY

dBASE to C translator: dbx -  
no royalties, addon ISAM.  
Library Pioneer it MSDOS \$ 350  
Source \$1000

## OTHER LANGUAGES

APL + PLUS/PC \$469  
ED/ASM-86 by Oliver Computing.  
Integrated editor/assembler/debugger  
w/8087 support. MSDOS \$ 85  
HS/FORTH - '79 & '83 Standards, full  
RAM, ASM, BIOS, interrupts, graph,  
multi-task, optimizer MSDOS \$250  
Ideal APL PCDOS \$469  
MacASM - fast \$99  
MasterForth by MicroMotion - floating  
point and relocator extensions  
available: Call MAC or PCDOS \$125  
MS MASM - improved MSDOS \$109  
Mystic Pascal - fast MSDOS \$ 64  
Paragon PASCAL - for performance:  
extensions like "packages", "Iterators",  
5 memory models. 64 bit 8087 strings.  
Space vs. speed optimization options.  
MSDOS \$895  
PASM - by Phoenix MSDOS \$239  
PL1-86 - Ansi subset \$499  
Prospero Pascal - full ISO + MSDOS \$390  
Turbo Edit ASM \$ 85

## XENIX-86 & SUPPORT

Basic - by Microsoft \$295  
Cobol - by Microsoft \$895  
Fortran - by Microsoft \$429  
Xenix Complete Development System Call

## OTHER PRODUCTS

CPRINT - by ENSCO \$ 50  
DoubleDOS - concurrent Call  
Faster C - scrap your linker \$ 95  
HTest/HFormat - thorough XT Fix \$119  
Microsoft Windows \$ 75  
Norton Utilities \$ 69  
Opt Tech Sort - sort, merge MSDOS \$ 85  
Polymake - Directly execute or GEN a  
batch file, batch, interactive.  
MSDOS \$129  
Texsys - control source MSDOS \$ 89

Note: All prices subject to change without notice.  
Mention this ad. Some prices are specials. Ask about  
COD and POs. All formats available.

Call for a catalog, literature, advice and service you can trust

## NEW HOURS

8:30 AM - 8:00 PM EST.

800-421-8006

THE PROGRAMMER'S SHOP™

128-D Rockland Street, Hanover, MA 02339  
Mass: 800-442-8070 or 617-826-7531 1285

"We at Sunspot are thrilled to know  
that there is a store that can cut through  
all the "bull", and find us the products  
that most computer stores know nothing  
about. Keep up the good work."

— Arland Hensler  
Sunspot

# The Great CRC Mystery

by Terry Ritter

**T**he Cyclic Redundancy Check (or CRC) is a way to detect errors in data storage or transmission. With more and more data being transmitted over phone lines, the need for protocols that protect data from damage in transit has increased, but the theory behind CRC generation is not well known.

## What Is a CRC?

The Cyclic Redundancy Check is a way to detect small changes in blocks of data. Error detection is especially important when computer programs are transmitted or stored because an error of even one bit (perhaps out of hundreds of thousands) is often sufficient to make a program faulty. Although a few errors in a text file might be acceptable (because the text can be reedited when received or recovered), an error-free file is preferable. An error-correcting protocol triggered by CRC error detection can provide this accuracy at low cost.

The CRC algorithm operates on a block of data as a unit.<sup>1</sup> We can understand the CRC better if we see this block of data as a single (large) numerical value. The CRC algorithm divides this large value by a magic number (the CRC polynomial or generator polynomial), leaving the remainder, which is our CRC result.

***Some implementations of XMODEM use a CRC algorithm that introduces unnecessary delay in data transmission. CRCs can be used in communications, startup verification of ROM code, and program and data correctness validation.***

The CRC result can be sent or stored along with the original data. When the data is received (or recovered from storage), the CRC algorithm can be reapplied and the latest result compared to the original result. If an error has occurred, we will probably get a different CRC result. Most uses of the CRC do not

attempt to classify or locate the error (or errors) but simply arrange to repeat the data operation until no errors are detected.

## Using the CRC

The IBM 8-inch floppy disk specification used the CRC-CCITT polynomial for error detection, and this CRC is now used in almost all floppy disk controller devices. A disk controller computes a CRC as it writes a disk sector, and then it appends that CRC to the data. When the data is read back, a new CRC is computed from the recovered data and compared to the original CRC. If the CRC values differ, an error has occurred and the operation is repeated. The standard disk CRC (CRC-CCITT) is hidden in the controller and nowadays receives little comment.

One version of the XMODEM (or Christensen) file transmission protocol also uses the CRC-CCITT polynomial to detect data transmission errors typically caused by line noise. When the receiving end detects a data error, it sends a NAK (Negative Acknowledge) character to the sender, which requests that the defective data block be retransmitted. The receiving end repeats this process un-

til the CRC from the transmitting end matches the local result or until one or both ends give up. When the result does match, the receiving end sends an ACK (acknowledge) character, and the transmitting end then sends the next block.

### Error Control and Efficiency

Many different CRC polynomials are possible; these generator polynomials are designed and constructed to have desirable error-detection properties. If the CRC polynomials are well constructed, the major difference between them is in their length. Longer polynomials provide more assurance of data accuracy and are fully usable over larger amounts of data; however, longer polynomials produce longer remainder values, adding error-checking overhead to the data.

A "16-bit" polynomial has a 16-bit remainder. There are two well-known 16-bit polynomials: CRC-16 (used in early BISYNC protocols) and CRC-CCITT (used in disk storage, SDLC, and XMODEM CRC). Of the two, CRC-CCITT may be a little stronger and by convention is often used in ways that strengthen its error-detection capabilities. This article illustrates CRC-CCITT, which is the pnomial  $x^{16} + x^{12} + x^5 + 1$ .

Polynomials are classified by their highest nonzero digit (or place), which is termed the degree of the polynomial. Both CRC-16 and CRC-CCITT are of degree 16, which means that bits 16 through 0 are significant in their description; a degree-16 polynomial thus has 17 bits. Normally we are most concerned with the remainder of the CRC operation, which has one bit less than the polynomial. Thus, we may think of 16-bit CRCs, even though their generator polynomials actually contain 17 bits (bits 16 through 0).

In a proper CRC polynomial, both the most significant bit (MSb) and least significant bit (LSb) are always a 1. Because the highest bit of the polynomial is always a 1, we are able to treat this bit differently from the other bits of the polynomial. Because the remainder from a sixteenth-degree polynomial has only 16 bits, a 16-bit register is sufficient for CRC operations on a 16-bit polynomial, even though the polynomial itself actually has 17 bits.

A well-constructed CRC polynomial over limited-size data blocks will detect any contiguous burst of errors shorter than the polynomial, any odd number of errors throughout the block, any 2 bit errors anywhere in the block, and most other cases of any number of errors anywhere in the data.<sup>2</sup> So every possible arrangement of 1, 2, or 3 bit errors will be detected. Nevertheless, there remains a small possibility that some errors will not be detected. This happens when the pattern of the errors results in a new value that, when divided, produces exactly the same remainder as the correct block. With a properly constructed 16-bit CRC, an average of one error pattern will not be detected for every 65,535 that would be detected. That is, with CRC-CCITT, we can detect 99.998 percent of all possible errors.<sup>3</sup>

There is no technique we can use to absolutely guarantee detection of any error, but we can minimize undetected errors at reasonable cost. Other error-detection techniques are available, such as checksum or voting, but these have poorer error-detection capabilities. For exam-

ple, the single-byte checksum (used in the original version of XMODEM) appears to be about 99.29 percent accurate,<sup>4</sup> which seems pretty good. But for a single additional byte, the CRC technique is about 460 times less likely to let an error pass undetected. In practice, the difference is much greater because the CRC will detect all cases of the most common errors at the cost of a 2-byte CRC value in every block. For example, the XMODEM protocol sends data in 128-byte blocks; these blocks can be CRC error-checked with an additional 2 bytes—an error-check overhead of about 1.5 percent.<sup>5</sup>

### Polynomial Arithmetic

The CRC performs its magic using polynomial modulo two arithmetic. Polynomial arithmetic mod 2 allows an efficient implementation of a form of division that is fast, easy to implement, and sufficient for the purposes of error detection. (This scheme is not particularly useful for the division of common numbers.) Polynomial arithmetic mod 2 differs slightly from normal computer arithmetic, and it is generally the most confusing part of the CRC.

A polynomial is a value expressed in a particular algebraic form, that of

$$A_n * X^n + A_{n-1} * X^{n-1} + \dots + A_1 * X + A_0$$

Our common number system is an implied polynomial of base 10: Each digit means the value of that digit is multiplied by the associated power of 10. The base 2 or binary system of numeration is another form of the general polynomial concept. When we see a number, we think of it as a single value; we mentally perform the polynomial evaluation in the assumed base to get a single result. On the other hand, formal polynomials are considered to be a list of multiple separate units, and the existence or evaluation of an ultimate single value for the polynomial may not be important.

Because decimal arithmetic uses constant-base polynomials, all of us already know how to do polynomial arithmetic in a constant base (10); however, the polynomials used in CRC calculations are polynomials modulo two. By modulo two we mean that a digit can have only values 0 and 1. Of course, this is always the case with binary values, so you might well wonder what all the mumbo jumbo is about. The difference is this: A modulo polynomial has no carry operation between places;<sup>6</sup> each place is computed separately. We perform mod 2 operations logically, bit by bit; in mod 2, the addition operation is a logical exclusive-OR of the values, and mod 2 subtraction is exactly the same (exclusive-OR) operation.

Modulo arithmetic is used for CRCs because of its simplicity: Modulo arithmetic does not require carry or borrow operations. In computing hardware, the carry circuitry is a major part of arithmetic computation and is a major contributor to speed limitations. Of course, because we have both subtraction and exclusive-OR instructions available in most computer instruction sets, this advantage is less important for software implementations of CRC. Nevertheless, the simplicity of modulo arithmetic allows several different software approaches not available in conventional arithmetic. Note that the modulo-type operations available in programming lan-

guages (e.g., the Pascal MOD operator) operate on entire numbers rather than individual bits or places.

A polynomial division mod 2 is very similar to common binary division, except that we perform a logical exclusive-OR operation instead of an arithmetic subtraction. Similarly, because "greater than" and "less than" are meaningless in modulo arithmetic, we can replace these operators by performing the exclusive-OR operation if the high bit is set or 1, driving the high part of the dividend to zeros.

We can implement a polynomial division as follows: A polynomial division register of a length corresponding to the remainder produced by the polynomial to be used is set up (see Figure 1, below).<sup>7</sup> Each element of the register should be able to hold the maximum modulo value; in mod 2, a single bit suffices. (Note that the hardware diagrams are intended only as examples. Very short CRCs are of limited practical use, and there are better ways to do the job.)

The register is cleared, then the data is shifted into the register from the right; each shift is a polynomial multiplication. Each shift also shifts a bit out of the register from the most significant bit (MSb). We know that the register value will exceed our representation when the shifted-out bit is logical 1, so we arrange to perform our polynomial subtraction when this happens; that is, when we shift out a 1, we exclusive-OR the polynomial with the value in the register. Because our polynomial (the magic number) always contains a high-order bit, which always forces the shifted-out bit back to a logical 0, we need not actually operate on the high-order bit. So only zeros shift out, keeping the mod 2 polynomial remainder in the register.

This bit-level hardware process is easily simulated. Turbo Pascal algorithms for the simulation are shown in

Listing One, page 76. Software simulation has the advantage of a fast and easy investigation of an algorithm, allowing quick changes to try out various forms of optimization. The program produces a "trace" of the execution, showing the step-by-step operation.

The polynomial division register does not hold the desired remainder until the place containing the last data bit has been shifted out of the register. To do this, a zero data bit must be shifted in for every bit of the register. In the case of CRC-CCITT, 16 bits (2 bytes) of zeros need to be appended to the data. After entering the zero bits, the result in the polynomial division register is the CRC result. The common implementations of XMODEM usually require these two trailing bytes.

The CRC result can be obtained without shifting in the two zero bytes by rearranging the CRC register and feeding the data in at the "top end" of the system (see Figure 2, below). By shifting the CRC register, we can shift zeros in from the right. The data bit will be compared to the MSb in the CRC register, and only if they differ will the polynomial be subtracted. As before, this acts to keep the full remainder in the register; however, the remainder is now correct after each bit and requires no trailing zeros. A simulation of this immediate-result algorithm (called, for lack of a better name, the CRC algorithm) is also given in Listing One for comparison to polynomial division. Notice that both the polynomial division and CRC algorithms come up with the same remainder (or CRC value), but the CRC version does it faster and with more consistent logic.

### Faster CRCs in Software

The bit-by-bit form of the CRC algorithm can be, and often is, directly simulated in software. The shifting and looping required by this approach can be reduced in several ways. Both byte-oriented<sup>8</sup> and table-oriented<sup>9</sup> algorithms have been available in the technical literature for a number of years. Table-oriented algorithms may (or may not) produce somewhat higher speed at the ex-

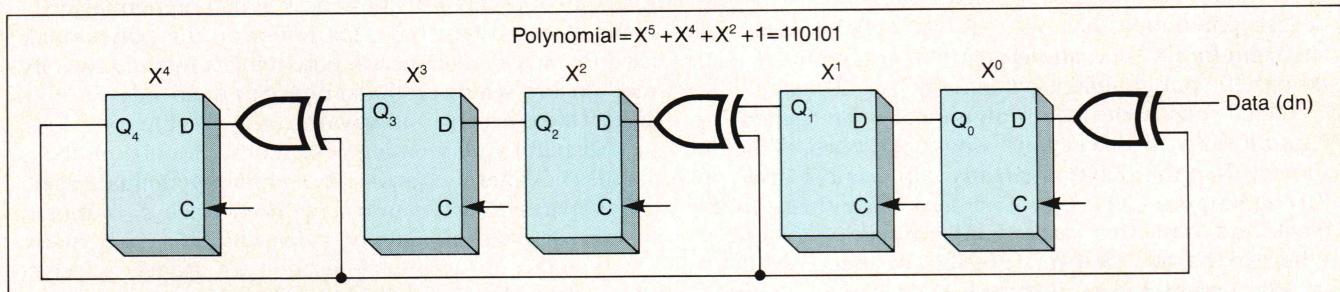


Figure 1: Polynomial Divide Hardware for a 4 bit CRC

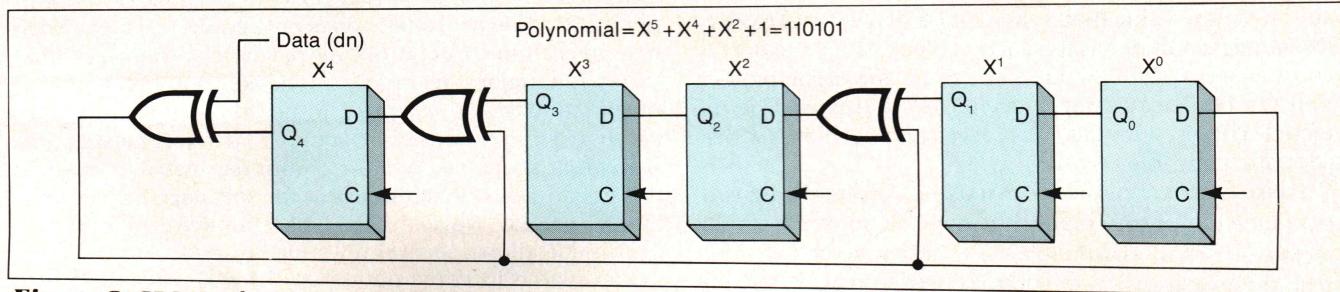


Figure 2: CRC Hardware

# C Programmers: Here are 8 ways You can be more productive

Dear Microcomputer Programmer,

Let me tell you how you can find and choose the best development software for your needs — software that will help you:

- \* Speed your development efforts
- \* Write even better programs

All you have to do is consider one of these eight products for C programmers (or the 97 other C compilers, interpreters, support libraries, debuggers, or addons we offer). Then call one of our knowledgeable consultants - toll free - for details, comparisons, or for our specially prepared packets on C, C Libraries, or C Productivity Tools.

There is no obligation. You risk nothing with our moneyback guarantee of satisfaction.

Yours for more productive programming,

— Bruce W. Lynch, President

## First Aid for C Programs C ToolSet

Save time and frustration when analyzing and manipulating C programs.

DIFF and CMP-for "intelligent" file comparisons.

XREF - cross references variables by function and line.

C Flow chart - shows what functions call each other.

C Beautifier - make source more readable.

GREP - search for patterns.

PP - formats your code so that it is easier to read and understand.

C Util - acts as a general purpose file filter.

There are several other programs for converting and printing programs.

Portable. Full source code. CPM, MSDOS \$135

## Even for Small Files: Convenient, Fast Access

CBTREE — Only \$99

Why spend time writing file management code when you can use consistent, flexible, documented, professional function? Even multiuser record locking and variable-length records are supported.

Full, balanced Btree support includes use of multiple keys, unlimited number and length of keys.

Use this powerful ISAM, even if you've previously done without.

Learn how to write systems for managing large files by using CBTREE source as a guide. Modify it and transfer it to another operating environment without royalties.

## SORT/MERGE Files for Clean, Fast Maintenance

with OPT-TECH SORT

Performance should not suffer with DOS or other "free" sorts. ISAMs alone are slow when 10% or even less is changed/added. OPT-TECH includes:

- CALLable and Standalone use
- C, ASM, BAS, PAS, F7N, COBOL
- Variable and fixed length
- 1 to 9 fields to sort/merge
- Autoselect of RAM or disk
- Options: dBASE, BTrieve files
- 1 to 10 files input
- No software max for # Records
- All common field types
- By pass headers, limit sort
- Inplace sort option
- Output = Record or keys

Try what you're using on an XT: 1,000 128 byte records, 10 byte key in 33 seconds. MSDOS \$85

## Add Communications Features to Your Programs

Greenleaf Comm Library

Greenleaf now enables you to communicate with remote systems or databases with an asynchronous communications library for C.

Individual transmission and reception ring buffers combine with an interrupt driven system. This eliminates the extra function of separately calling up the communications program.

Included are 1 library/object files, 100 functions; 100 page manual, complete source code, library tailor-made to suit compiler and memory. Hayes-compatible modem commands, and a complete sample file transfer program.

MSDOS \$149

## Get File Access with TIGHTER Control

db\_VISTA Data Management

Full source, no royalties and "normal" indexed file management are part of db\_VISTA. Get more for the price of only an ISAM.

You can minimize data stored and access records even faster and more logically than just using indexes. Example: address and transaction data should not require redundant storage of customer names or numbers. Use pointers. Related data fields point to other related groups - the "network model" of data.

Use db\_VISTA as a "normal ISAM" or save programming time, access time and file size. Lattice, C86, Williams, Desmet, Microsoft C.

MSDOS Multiuser source \$995, Object \$495

Single user source \$450, Object \$169

Unix, Xenix, & Macintosh versions also available. Call for details.

## Shorten Development Time, Cut Frustrations BRIEF, The Programmer's Editor

Compile within BRIEF; use autoindent; "templates", C specific error support... use windows, UNDO, and multiple large files.

But edit YOUR WAY.

Every feature you'd expect and more are integrated elegantly - and it can be modified by you.

You deserve:

"...the best text editor you can buy." - John Dvorak, InfoWorld,

7/8/85

"...the best code editor..." David Irwin, Data Based Advisor,

8/85

PCDOS \$Call

If you call for our advice, you must be completely satisfied with the product you purchase from The Programmer's Shop. If not, you will receive a refund or replacement. Call now for details or our new catalog.

## Make REAL TIME Programming Practical

Csharp Realtime Toolkit

Data acquisition, process control, robotics and devices monitoring applications become practical with Csharp!

Full source code helps tailor programs to various boards and applications.

Reentrant, interrupt handling routines help schedule and react. Fast graphics routines help visualize what is happening.

Control multiple ports reliably, schedule tasks based on events, manage priorities — all with modular, tested, and reliable routines.

Assess and manage the state of hardware at the object level. Let Csharp handle the details.

Portable C source supports RT11 UNIX and MSDOS \$600

## Fast File Access with Source Variable Length Fields Save Space

CIndex ISAM Product Line

C-Index contains a high performance ISAM, balanced B + Tree indexing system and *variable length* fields. The result is a complete data storage system to eliminate tedious programming and add efficient performance to your programs.

Features include random and sequential data access, virtual memory buffering, and multiple key indexes.

With no royalties for programs you distribute, full source code, and variable length fields C-Index/Plus fits what you are likely to need.

Save time and enhance your programs with C-Index/Plus. MSDOS \$349. With C-Index/File for \$89, or/Pro for \$179.

## THE PROGRAMMER'S SHOP

128 Rockland Street  
Hanover, Massachusetts 02339

800-421-8006

In Mass. 800-442-8070 617-826-7531

pense of a sizable table of constants that generally must be initialized before use. Examples of the various forms of CRC algorithms are given in Listing Two (page 78).

We can speed up the algorithm even more by precomputing the CRC for all possible combinations of a 16-bit CRC and a data byte and then saving the results. Done naively, this would be a transformation of 24 bits (16 bits of the previous register and 8 bits of data) into 16 bits. This approach would thus require  $2^{25}$  bytes (about 34 megabytes) of lookup table. In order to make the table approach practical, we must find a way to reduce the size of the table.

If we examine the CRC hardware, we notice that the current data bit is always combined with the current MSb of the CRC register. When we compute a whole-byte CRC, we end up combining the whole data byte with the high byte of the CRC. We can precompute the exclusive-OR of the data byte and the high byte of the CRC register (this is a single-byte operation in software), yielding a single byte we can call the combined term or the combination value.

For the common 16-bit CRCs, it turns out that the CRC register changes in patterns that have a direct mapping from the combination value. Thus, it is possible to precompute the CRC changes for all 256 possible combination values. Then, when we need to do a CRC, we can use the 1-byte combination value to look up a corresponding 2-byte result, then use that result to correctly change the CRC register. As you might expect, the required change is simply a 2-byte exclusive-OR operation.

To generate the data for the lookup table, we need only generate the 2-byte CRC result for all 256 possible data bytes, given an "all-zeros" starting CRC register. Each result is a 1 for those bits in the CRC register that are changed by a particular combination code. We can use a nontable implementation of the CRC to compute the table values.

This approach to generating a table of CRC values requires a 512-byte lookup table. We must fill the table with the correct data in an initialization step and perform a few more run-time operations than the straight lookup process requires (compute the combination value, look up the result, then apply the result to the CRC register and compute the new CRC value).

Another variation that is faster than the original bit-by-bit approach and that also eliminates the lookup storage of the table approach is the bytewise shifting algorithm. A bytewise approach eliminates seven bit-by-bit test-and-jump operations, which are a significant overhead in the bit-by-bit version, and also takes advantage of fast-shift and parallel-logic operations available on most processors (as well as some high-level languages such as Turbo Pascal or C).

First, we need some more algebra: By giving each CRC register bit and each data bit a separate symbol, we can express the result of a CRC operation symbolically. Each bit of the CRC register will be represented by a formula showing all the data and original CRC bits that affect that bit in the result. If we take the exclusive-OR of the bits

specified by the formula, we can directly calculate any bit of the CRC result.

In order to generate the formulas for each bit of the CRC register, we create an algebraic analog of the shifting and combining process of the bit-by-bit CRC algorithm. Instead of shifting bit values (as in a normal shift register), we move the whole symbolic formula for each bit to the next higher bit position. Instead of actually performing an exclusive-OR operation, we concatenate the formula for the data bit to each of the affected bits in the CRC register with a symbol indicating an exclusive-OR operation. If ever we find two identical variables in any one formula, we can cancel and eliminate them both (because anything exclusive-ORed with itself is zero, and zero exclusive-ORed with any value is just that value).

After symbolically processing a whole byte of data and eliminating common terms, we come up with a symbolic representation for each bit of the result. By factoring this expression into convenient computer operations, a program is obtained that uses the bit parallelism available in software.

### CRC Deviations

More improvement is possible. We have previously assumed that the CRC register is cleared before starting the computation and also that we specifically compare the stored (or transmitted) CRC value to the current CRC result. These assumptions are discarded in protocols other than XMODEM.<sup>10</sup>

When a CRC register contains only zeros, processing a zero data bit does not change the CRC remainder. So, if the CRC register is clear and extraneous zero bits do occur, these data errors will not be detected. For this reason, most current CRC protocols initialize the CRC register to all 1s before they start the computation, allowing the detection of extraneous leading zeros.

We can also eliminate the need to detect the separate CRC field at the end of a data block. If the CRC result is simply attached to the end of the data, the receiving CRC register will clear itself automatically if there is no error; that is, each bit of the stored or transmitted CRC value should cancel the similar bit in the CRC register. Although of minor importance for software implementations, this is a reasonable simplification for hardware CRC devices because it allows the same hardware to be used regardless of block length.

When the CRC remainder is appended to the end of the data (thus eliminating the need to detect it as a separate field), and if bit-level CRC hardware is also to be supported, CRC software may need to use data in reverse bit order. This is because bit-level CRC hardware works on data after it has been serialized, and data is traditionally serialized LSb-first. That is, the parallel-to-serial conversion in an asynchronous serial device sends the rightmost bit of a character first and the leftmost bit last. The bit-level CRC hardware has little choice but to treat the resulting data-stream as a single large number, but that data-stream has its byte-level bit-order changed from our usual numerical expectations.

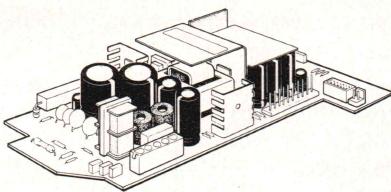
If an MSb-leftmost CRC routine is to be compatible with bit-level CRC hardware, it may be necessary to reverse the bit order of every data byte (before each is processed

# DIGITAL RESEARCH COMPUTERS

(214) 225-2309

## Switching Power Supply!

- + 5VDC - 8 Amps
- +12VDC - 5 Amps
- 12VDC - 1 Amp
- (81 WATTS MAX)



**\$29.95**  
ea.

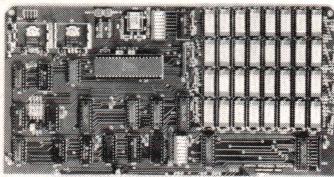
4 FOR \$99

ADD \$2  
EA. UPS

**BRAND NEW UNITS, MFG. BY BOSCHERT FOR HEWLETT PACKARD! PERFECT FOR SMALL COMPUTER AND DISK DRIVE APPLICATIONS #XL81-5630. INPUT 110V/220V, 50/60 HZ. NOMINAL OUTPUTS: +5VDC @ 8A, +12VDC @ 5A, -12VDC @ 1A. TOTAL MAX OUTPUT. MUST BE LIMITED TO 81 WATTS TOTAL! (WITH PIN OUT SHEET.) ORIGINAL FACTORY BOXED.**

**256K S-100 SOLID STATE DISK SIMULATOR!**  
WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

**PRICE CUT!**



BLANK PCB  
(WITH CP/M 2.2  
PATCHES AND INSTALL  
PROGRAM ON DISKETTE)  
**\$69.95**  
(8203-1 INTEL \$29.95)

**\$149.00**  
(ADD \$50 FOR A&T)  
#LS-100  
(FULL 256K KIT)

**FEATURES:**

- \* 256K on board, using + 5V 64K DRAMS.
- \* Uses new Intel 8203-1 LSI Memory Controller.
- \* Requires only 4 Dip Switch Selectable I/O Ports.
- \* Runs on 8080 or Z80 S100 machines.
- \* Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
- \* Provisions for Battery back-up.
- \* Software to mate the LS-100 to your CP/M® 2.2 DOS is supplied.
- \* The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
- \* Compare our price! You could pay up to 3 times as much for similar boards.

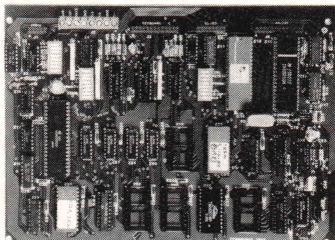
### THE NEW ZRT-80

## CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

**FEATURES:**

- \* Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
- \* RS232 at 16 BAUD Rates from 75 to 19,200.
- \* 24 x 80 standard format (60 Hz).
- \* Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
- \* Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
- \* Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
- \* 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
- \* Composite or Split Video.
- \* Any polarity of video or sync.
- \* Inverse Video Capability.
- \* Small Size: 6.5 x 9 inches.
- \* Upper & lower case with descenders.
- \* 7 x 9 Character Matrix.
- \* Requires Par. ASCII keyboard.



**\$89.95**  
(COMPLETE KIT, 2K VIDEO RAM)

BLANK PCB WITH 2716  
CHAR. ROM, 2732 MON. ROM

**\$49.95**

SOURCE DISKETTE - ADD \$10

SET OF 2 CRYSTALS - ADD \$7.50

FOR 8 IN. SOURCE DISK  
(CP/M COMPATIBLE)  
ADD \$10

## Digital Research Computers

P.O. BOX 381450 • DUNCANVILLE, TX 75138 • (214) 225-2309

**64K S100 STATIC RAM**  
**\$119.00**  
KIT

**LOW POWER!**

150 NS ADD \$10

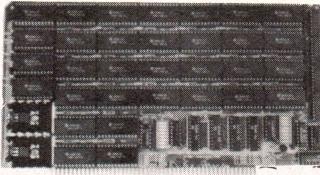
BLANK PC BOARD  
WITH DOCUMENTATION  
**\$49.95**

SUPPORT ICs + CAPS  
**\$17.50**

FULL SOCKET SET  
**\$14.50**

FULLY SUPPORTS THE  
NEW IEEE 696 S100  
STANDARD  
(AS PROPOSED)  
FOR 56K KIT \$105

ASSEMBLED AND  
TESTED ADD \$50



**FEATURES: PRICE CUT!**

- \* Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- \* Fully supports IEEE 696 24 BIT Extended Addressing.
- \* 64K draws only approximately 500 MA.
- \* 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- \* SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
- \* 2716 EPROMs may be installed in any of top 48K.
- \* Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
- \* Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
- \* BOARD may be partially populated as 56K.

## PANASONIC Green Screen - Video Monitors

25 MHZ. TYPICAL BANDWIDTH!!!

Brand New In The Box! 9-Inch Screen

#K-904B1 (Chassis #Y08A) Open Frame Style

**\$29.95**

EA.

GROUP SPECIAL:  
**4 for \$99.00**  
WITH DATA & SCHEMATIC

(USA SHIPPING: \$3. PER UNIT. CANADA: \$7. PER UNIT)

COMPUTER MANUFACTURER'S EXCESS. STILL IN ORIGINAL PANASONIC BOXES. THE CRT TUBE ALONE WOULD COST MORE THAN OUR PRICE FOR THE COMPLETE UNIT. FOR SPLIT VIDEO (TTL INPUTS) OPERATION, NOT COMPOSITE VIDEO. OPERATES FROM 12VDC AT 1 AMP. VERTICAL INPUT IS 49 TO 61 HZ. HORIZONTAL INPUT: 15,750 HZ ± 500 HZ. RESOLUTION IS 800 LINES AT CENTER 650 LINES AT CORNERS.

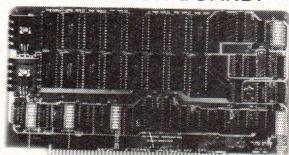
## 32K S100 EPROM/STATIC RAM

**NEW!**

FOUR FUNCTION BOARD!

**NEW!**

EPROM II  
FULL  
EPROM KIT  
**\$69.95**  
A&T EPROM  
ADD \$35.00



BLANK  
PC BOARD  
WITH DATA  
**\$39.95**

SUPPORT  
ICs  
PLUS CAPS  
**\$16**

FULL  
SOCKET SET  
**\$15**

We took our very popular 32K S100 EPROM Card and added additional logic to create a more versatile EPROM/RAM Board.

**FEATURES:**

- \* This one board can be used in any one of four ways:
  - A. As a 32K 2716 EPROM Board
  - B. As a 32K 2732 EPROM Board (Using Every Other Socket)
  - C. As a mixed 32K 2716 EPROM/2K x 8 RAM Board
  - D. As a 32K Static RAM Board
- \* Uses New 2K x 8 (TMM2016 or HM6116) RAM's
- \* Fully Supports IEEE 696 Buss Standard (As Proposed)
- \* Supports 24 Bit Extended Addressing
- \* 200 NS (FAST!) RAM's are standard on the RAM Kit
- \* Supports both Cromemco and North Star Bank Select
- \* Supports Phantom
- \* On Board wait State Generator
- \* Every 2K Block may be disabled
- \* Addressed as two separate 16K Blocks on any 64K Boundary
- \* Perfect for MP/M® Systems
- \* RAM Kit is very low power (300 MA typical)

**32K STATIC RAM KIT — \$99.55**

For RAM Kit A&T — Add \$40

TERMS: Add \$3.00 postage. Orders under \$15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over \$50 add 85¢ for insurance.

or serialized) and also the CRC remainder bytes (after the block ends). Bit-order reversal can be done in software, hardware, or both. Alternately, the CRC algorithm could be reconstructed so as to use and hold MSb-rightmost data.

In strictly software CRC implementations, however, we work on data before it is serialized and after it is recovered, and we trust any serialization that occurs to be transparent. We can't afford to treat the data-stream as a single large value with MSb-leftmost, with MSb-leftmost bytes, and a similar MSb-leftmost CRC remainder appended on the right. This arrangement is most consistent with both the theory and our numerical conventions and is the form used by XMODEM. The CRC routines shown in this article use MSb-leftmost data and keep the result also in MSb-leftmost format.

If we arrange to verify the CRC by processing the CRC result as data, we again fall prey to extraneous zero data bits. In order to detect such errors, we arrange for the CRC register to take on a unique nonzero value in the event of no error. By some quirk of the algebra, it turns out that if we transmit the complement of the CRC result and then CRC-process that as data upon reception, the CRC register will contain a unique nonzero value depending only upon the CRC polynomial (and the occurrence of

no errors). This scheme is now used by most CRC protocols, and the magic remainder for CRC-CCITT is \$1D0F (hex).

### Actual CRC Implementations

I constructed several CRC implementations for speed and size comparisons (see Listing Two). The CRC-CCITT polynomial was used because this is the polynomial used in XMODEM, as well as many other data communication uses. I used Turbo Pascal, though the code could obviously be rewritten in C. A couple of the operations used are Turbo Pascal extensions: *Swap()* is an INTEGER function that exchanges the high and low byte of an integer value; *Lo()* is an INTEGER function that selects only the low byte of an integer.

I used the Pascal Bit-by-Bit approach (a direct simulation of the hardware method) to provide a reference against which the other algorithms are compared. The Pascal Fast B-B-B is an improved bit form comparable to most high-level language implementations of the XMODEM CRC, except that this version requires no trailing zeros to finish the calculation (and so is already faster than the usual version). The Pascal Byte version illustrates the improvement wrought from algebraic factoring; the Pascal Table version shows how a precomputed table can simplify and speed execution-time operation. The Machine Code versions of Byte and Table show yet more improved speed. The different approaches illustrate various trade-offs of speed, space, and specialization. The results (Table 1, left) show a range of almost two orders of magnitude in execution speed.

Each CRC implementation was made into a Pascal PROCEDURE for easy testing and comparison. For validation, varying amounts of program code from main memory were processed by each implementation. All algorithms achieved the same results. Several of these versions have been placed in an implementation of XMODEM with good results.

### Time Tests

For the time tests, each implementation was executed 10,000 times under Turbo Pascal 3.01A on an 8088 in a Leading Edge PC with a 7.16-megahertz (MHz) clock; the times would be 50 percent longer on an IBM PC. The time was taken automatically from MS DOS. Because the MS DOS timer ticks only about 18.2 times per second, this method is only precise within about 55 milliseconds (msec) at both the start and end of the timing interval. The large number of repetitions minimize this effect.

The time reported as "10,000 uses" is real time decreased by the amount of time taken by 10,000 empty loops, thus giving us the time associated with the procedure call and execution instead of also including the looping structure that we use only for the tests. The In Line column decreases "10,000 uses" by the time taken for 10,000 procedure calls and returns, giving the time for execution only.

### Selection Criteria

The time necessary to process a byte (including the CRC operation and whatever queuing operations and other tests that need to be performed) should be less than the

CRCTIME, 85/9/18  
Execution times for various CRC implementations.  
Copyright (c) 1985, T.F. Ritter; All Rights Reserved.

#### BEGIN Validation Testing.

Reference = crcitby (Pascal Bit-By-Bit).  
crcfbbb (Pascal Fast Bit-By-Bit): No error.  
crcitta (Pascal Byte): No error.  
crcatabl (Pascal Table): No error.  
mrcrcitt1 (Machine Code Byte): No error.  
mrcrcitt3 (Machine Code Table): No error.

#### END Validation Testing.

Turbo Pascal runs CRC-CCITT on 8088 under Bare MSDOS  
FOR 10000 OPERATIONS; 7.16 MHz CLOCK (multiply by 1.5  
for 4.77 MHz)

Empty loop: 0.160 secs

Empty procedure in loop: 0.880 secs  
(procedure overhead alone = 0.072 msec each)

	10,000 Uses(secs)		1 Use (msec)	
	Procedure	In Line	Procedure	In Line
Pascal Bit-by-Bit:	13.790	13.070	1.379	1.307
Pascal Fast B-B-B:	7.310	6.590	0.731	0.659
Pascal Byte:	2.150	1.430	0.215	0.143
Pascal Table:	1.430	0.710	0.143	0.071
Machine Code Byte:	1.050	0.330	0.105	0.033
Machine Code Table:	0.890	0.170	0.089	0.017

Table 1

Free CompuView Software!  
Call (313) 996-1299 for details.

**VEDIT PLUS** ®

**Multiple File Editor**

**Word Processor**

# RECOMMENDED

Come and get it. The editor you've heard so much about. The editor that has been recommended to you for the last five years by BYTE, InfoWorld, Microcomputing, PC, Programmers Journal, Sextant, EDN, Jerry Pournelle, Peter Norton and numerous other reviewers.

*'VEDIT is a lightning fast text editor with all the commands of a slick word processor...a text oriented programming language enables you to perform tasks impossible with a standard word processor. A fantastic product.'*

#### **The Whole Earth Software Catalog, 1985.**

'VEDIT is fast, functionally rich and configurable to your whims. Its programming ability lets its usage stretch as far as your imagination will allow.'

#### **The OMNI Complete Catalog of Computer Software, 1985.**

Now there's new VEDIT PLUS. It does it all. Program development. Document preparation. Word Processing. Convert WordStar files, edit dBASE source files, process mainframe files. Whatever your application, VEDIT PLUS can handle it.

VEDIT PLUS is the choice of thousands of engineers, writers and programmers who demand the most powerful text editing software.

Expect more from VEDIT PLUS. It's available for all MS-DOS, PCDOS, CP/M-86 and CP/M-80 computers. List price \$225. Educational / corporate site licensing and current user discounts available.

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. WordStar is a registered trademark of MicroPro, International. dBASE is a registered trademark of Ashton Tate. MS-DOS is a registered trademark of Microsoft. CP/M is a registered trademark of Digital Research.

#### **VEDIT PLUS FEATURES**

- Simultaneously edit up to 37 files of unlimited size. Do 'cut and paste' operations, edit text or develop macros.
- 'Virtual' disk buffering simplifies editing of large files.
- Memory management supports up to 640K
- MS-DOS, PCDOS pathname and CP/M user number support.
- Horizontal scrolling (edit spreadsheets easily).
- Customization - determine your own keyboard layout, support any screen size, any computer, any CRT terminal

#### **EASY TO USE**

- Interactive on-line help. Create your own on-line help with menus for compilers, style guides, non-computer subjects.
- On-Line integer algebraic calculator.
- 'Undo' command.
- Single key search and selective replace. Search with wild cards and pattern matching.
- Keystroke macros - create your own on-line editing functions.
- Print any portion of text or file.
- Excellent indexed documentation. Includes new tutorial.
- Highly optimized for IBM PC, PC XT or PC AT.

#### **PROGRAM DEVELOPMENT**

- Automatic Indent/Indent for 'C', PL/I or PASCAL.
- Conditional lower to upper case conversion. Change at ';' or other character.
- Optional 8080 to 8086 source code translator macro.

#### **WORD PROCESSING**

- Word wrap and paragraph formatting at adjustable margins.
- Right margin justification.
- Recognizes graphic, foreign and special character sets.

#### **MACRO PROGRAMMING LANGUAGE**

- 'IF-THEN-ELSE', looping, conditional branching, user prompts, keyboard input, algebraic expressions and variables.
- Simplifies complex text processing, formatting, conversions and translations.
- Edit multiple files automatically - perform numerous search/replace in several files without user intervention.
- Complete TECO capability.
- Free Macros: • Compare two files and merge work done by two people • Sort mailing lists • Print formatter • Replace command prompt with an easy to use menu.

time it takes to receive a character. We could simply accumulate the data in a block as it is received and then CRC-process the whole block, but this procedure would add some delay, or latency, between receiving the last data byte and returning a response to the sender (ACK for good data and NAK for an error in XMODEM). Some XMODEM implementations appear to use this method, giving the impression that the protocol or the CRC is responsible for the delay. Because fast CRC routines are obviously possible, it is hard to rationalize any latency at all.<sup>11</sup>

The Pascal Byte version, which takes only a few lines of code and is machine independent (under Turbo Pascal), may be suitable for speeds up to 9,600 bps and is a reasonable choice for most use. The Pascal Table version is a little faster, but the table generally must be initialized before use, either by using a different CRC version, or perhaps by reading the values in from a file. Alternately, (in most languages) the table could be defined in the source code as a large body of constants.

The faster versions can generally benefit from being used in-line (that is, not as procedures) to avoid procedure call/return overhead, but this is also inconvenient because each use would involve duplicating the same code in different places. The Machine Code Table version is shorter and so would minimize the duplication penalty. The Pascal Table version can also be used in-line because it takes a minimum amount of code. I use an Include file holding the Machine Code Byte version, then call the routine as a procedure; the resulting code is both small and fast.

### Other Uses

Although this article has concentrated on CRCs in communications and data storage, CRCs can be used in many different applications involving error detection. Such applications include start-up verification of ROM code, load-time verification of RAM modules (as in the 6809 operating system OS9), and program and data correctness validation.

Note that CRC polynomials are designed and constructed for use over data blocks of limited size; larger amounts of data will invalidate some of the expected properties (such as the guarantee of detecting any 2-bit errors). For 16-bit polynomials, the maximum designed data length is generally  $2^{15} - 1$  bits, which is just one bit less than 4K bytes. Consequently, a 16-bit polynomial is probably not the best choice to produce a single result representing an entire file or even to verify a single EPROM device (which is now commonly 8K or more). For this reason, the OS9 polynomial is 24 bits long.

### How To Learn More

A good introduction to CRCs can be found in the classic *Error Correcting Codes*, 2d ed., by Peterson and Weldon (Cambridge, Mass.: MIT Press, 1972), but you can expect to do some serious math to understand it. A brief nonmathematical chapter on CRC error detection in data applications (with some good figures) is available in *Technical Aspects of Data Communication*, 2d ed., by J. McNamara (Digital Equipment Corporation: Digital Press, 1982). The

very brief section in *Computer Networks* by A. Tanenbaum is also fairly good.

### Notes

1. The CRC does not require a fixed block size (though there is a built-in maximum), but some error-correcting protocols do. Larger amounts of data are simply partitioned into blocks that are considered separately.
2. Peterson, W. W., and Brown, D. T. "Cyclic Codes for Error Detection." *Proceedings of the IRE* (January 1961): 228-35.
3. Tanenbaum, A. *Computer Networks*. 128-32. Englewood Cliffs, N.J.: Prentice-Hall, 1981.
4. Brooks, L., and Rasp, J. "How Accurate is Accurate?" *DDJ* (February 1984): 27.
5. Error detection is only part of the requirements for a protocol. Other requirements include transmitting the data in blocks, numbering the blocks, and responding when a block has been received. The corresponding design decisions in XMODEM typically add yet another four bytes to each block transferred, for a required overhead of about 4.5 percent. This value can be, and often is, additionally degraded in implementation.
6. The general case of polynomial arithmetic, which allows a nonconstant base, generally makes carry operations (between terms) difficult.
7. It is common and traditional for the CRC register to be shown shifting right, which is the exact reverse of this author's analogy to binary division. Given our system of numeration, it seems reasonable to place most significant digits of one value to the left, and it is then correct for the CRC register to be seen as shifting to the left.
8. Helness, K. "Implementation of a Parallel Cyclic Redundancy Check Generator." *Computer Design* (March 1974): 91-96; Vasa, S. "Calculating an Error-Checking Character in Software." *Computer Design* (May 1976): 190-92; Socha, H., et al. "Letter to the editor." *Computer Design* (May 1979): 6, 12; Kjelberg, I. "Letter to the editor." *IEEE Micro* (August 1985): 4, 99.
9. Whiting, J. "An Efficient Software Method for Implementing Polynomial Error Detection Codes." *Computer Design* (March 1975): 73-77; Perez, A. "Byte-wise CRC Calculations." *IEEE Micro* (June 1983): 40-50; Schwaderer, D. "CRC Calculation." *PC Tech Journal* (April 1985): 118-32.
10. McKee, H. "Improved CRC Technique Detects Erroneous Leading and Trailing 0's in Transmitted Data Blocks." *Computer Design* (October 1975): 102-6; Fortune, P. "Two-Step Procedure Improves CRC Mechanism." *Computer Design* (November 1977): 116-29.
11. Some protocols other than XMODEM allow subsequent blocks to be sent before a previous block is acknowledged, thus minimizing the latency problem.

DDJ

(Listings begin on page 76)

**Reader Ballot**  
Vote for your favorite feature/article.  
Circle Reader Service No. 202.

# BetterBASIC Version 2.0

## "I Wrote It, And I Recommend It."

*"BetterBASIC has evolved into a programming environment which is completely compatible with GW BASIC and PC BASICA when running on IBM PCs and compatibles. Now you can easily load your old BASIC programs into BetterBASIC. BetterBASIC gives programmers use of the full memory of the computer and a structured language with true procedures and functions—like PASCAL and C. I wrote BetterBASIC and I recommend it."*

Ivar Wold, President • Summit Software Technology Inc. • Norwood, MA

### ACCESS FULL MEMORY—

BetterBASIC accesses the full memory of the computer enabling you to overcome Microsoft's 64K barrier.

### INTERACTIVE

**COMPILER**—BetterBASIC compiles to an intermediate code giving you five to six times the speed of traditional BASICs. There is immediate feedback on line entry.

**COMPATIBLE**—Version 2.0 of BetterBASIC is GW-BASIC, PC-BASICA compatible when running on IBM

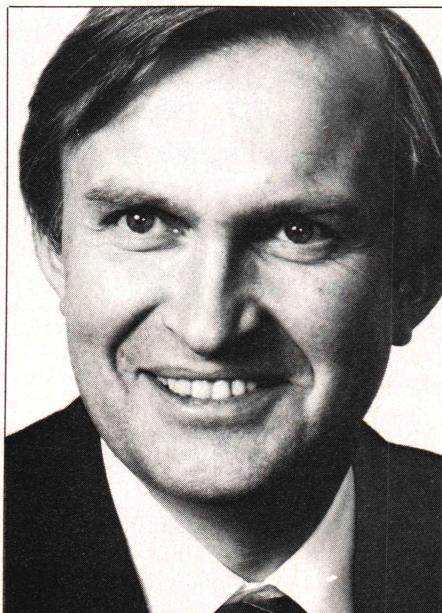
PCs. BetterBASIC is easy to learn because the syntax is the same.

**STRUCTURE**—Create well-organized programs using procedures and functions that are easily identified and understood.

**NOT COPY PROTECTED**—Install BetterBASIC on your hard disk. BetterBASIC is licensed to the programmer, so you can compute at work and at home using the same copy of BetterBASIC.

**USER DEFINED KEYWORDS**—The BetterBASIC language can be extended by adding your own procedures and functions to the language as keywords.

**RUNTIME SYSTEM**—Creates stand-alone EXE. files. Developers can distribute their programs written in BetterBASIC without royalties.



**SAMPLE DISK**—Contains a tutorial, a demo, and allows you to use an abbreviated form of BetterBASIC. It also contains a 60 page on-line mini manual.

**AND MORE**—Such as DOS and BIOS ROM calls, Chaining, Overlays, Local and Global Variables, Recursion—Graphics and Windows—You can define up to five windows. Optional 8087/80287 Math Chip Support.

**LIBRARIES**—Write reusable code.

### TECHNICAL SUPPORT—

Available to all registered users.

**BetterBASIC** Runs on IBM PC, XT, AT and all IBM-compatibles. Ask your local dealer for BetterBASIC or call 1-800-225-5800. In Canada call 416-469-5244. Also available for the Tandy 1000, 1200 and 3000 at Tandy/Radio Shack stores.

### PRICES:

Better BASIC	\$199
8087/80287 Math Module	\$99
Runtime System	\$250
Sample Disk with Tutorial	\$10

*Better*  
**BASIC.**

**Because It's The Best.**

Circle no. 228 on reader service card.

**Summit Software Technology, Inc.™**

106 Access Road, Norwood, MA 02062

MasterCard, Visa, Checks, Money Order, C.O.D. accepted and P.O. on approval.

BetterBASIC is a registered trademark of Summit Software Technology Inc.

IBM PC, XT, AT, are registered trademarks of International Business Machines Corp. Tandy is a registered trademark of Tandy Corp.  
(If you're using BetterBASIC and would like to be featured in one of our ads, please write to the Director of Advertising at Summit.)

# Fast Integer Powers for Pascal

by Dennis E. Hamilton

**B**ecause the Pascal language does not provide any built-in operation (such as FORTRAN's  $x^{**n}$ ) for powers of numbers, programmers are led to try the costly and unreliable substitute  $\exp(n * \ln(x))$ . When  $n$  is an integer, however, there are often ways to factor out computation of powers altogether. When direct use of powers is still desirable, the Pascal function  $\text{PowerN}(x, n)$  introduces the fastest method known for general computation of  $x^{**n}$  for  $n$ , an unrestricted integer not known in advance.

## Increasing Use of Powers

Repetitive computation of powers of numbers is almost commonplace now that personal computers and software make elaborate arithmetic computations more affordable and easier to express.

Powers do not occur merely in the evaluations of polynomials and the summations of series for computing specialized mathematical functions. Powers (especially ones with variable exponents) occur routinely in work with statistics and probabilities and in financial and economic formulas.

Typically, the exponent is an exact integer, opening up the possibility of using special shortcuts for faster and

***The function introduces the fastest method for computation of  $x^{**n}$ . This decade has seen renewed interest in computation of powers.***

more accurate computation. The Pascal  $\text{PowerN}$  function introduced in this article uses just such shortcuts to implement the fastest general computation known.

$\text{PowerN}$  is extremely efficient on systems, including most microcomputers, where floating-point arithmetic is more costly than small-integer hardware operations. Use of  $\text{PowerN}$  is also almost always preferable to use of the questionable  $\exp(n * \ln(x))$  logarithmic method. (See sources 1; 7; 15; 11, Section 4.6.4 for even better possibilities.)

## Powers Defined

For integer exponent value,  $n$ , the  $n$ th power of the base value  $x$  is designated here by the notation  $x^{**n}$  originally popularized in FORTRAN. These powers are usefully understood in terms of repeated-multiplication operations according to the scheme proposed in 1676 by Isaac Newton:

$$\begin{aligned} x^{**n} &= x * x * \dots * x, & n > 0 \\ &\quad \leftarrow n \text{ x's} \rightarrow \\ &= 1, & n = 0 \text{ and } x < > 0 \\ &= 1/x^{**(-n)}, & n < 0 \text{ and } x < > 0 \end{aligned}$$

This formulation of  $x^{**n}$  (and the mathematician's  $x^n$ ) satisfies our intuitions (for  $n > 0$  at any rate) and

allows use of some nicely generalized "laws of exponents" for powers:

$$\begin{aligned} x^{**}(i+j) &= x^{**i} * x^{**j}, \\ x < > 0 \text{ or } (i > 0 \text{ and } j > 0) \\ x^{**}(i*j) &= (x^{**i})^{**j}, \end{aligned}$$

The restrictions are significant: Standard mathematics defines no specific quotients for divisions by zero; it is technically important to avoid sneaking any in as loopholes of the exponent laws. With due allowance for the restrictions, satisfaction of these laws justifies the shortcuts in practical computations with variable, large  $n$ .

## Algorithm Performance

Pascal function  $\text{PowerN}(x, n)$ , given in Listing One, page 84, derives  $x^{**n}$  using exactly

$$\begin{aligned} \text{MC}(n) &= \text{floor}(\lg(n)) + \\ &\quad \text{bits1}(n) - 1, & n > 0 \\ &= 0, & n = 0 \\ &= \text{MC}(-n), & n < 0 \end{aligned}$$

multiplications. Here,  $\text{bits1}(n)$  is the number of 1-bits in the representation of  $n$  by a binary integer,  $\lg(n)$  is the base 2 logarithm of  $n$ , and  $\text{floor}(x)$  is the greatest integer that isn't larger than  $x$ . Even the ambitious case  $\text{PowerN}(x, 32767)$  is computed with only 28 multiplication operations (including  $\text{sqr}(x) = x * x$ ).

Another part of the claim to  $\text{PowerN}$ 's superiority, despite its superficial complexity, involves the decision cost expended in order to achieve so few multiplications. The number of questions of the form  $\text{odd}(i)? i < > 1? n < 0?$  that have to be asked is

$$\text{DC}(n) = \text{MC}(n) + 4, \quad n < > 0$$

where two are just for checking the

© 1985 by Dennis E. Hamilton, 1952 Baird Rd., Penfield, N.Y. 14526. Permission is granted for copying the software in this article for use in an electronic computer, whether or not for commercial advantage, provided that all notices of copyright and authorship appear and this publication and its issue date are credited. To copy or republish otherwise requires specific permission and may require payment of a fee.

special  $n = 0$  and  $n < 0$  cases. The two extras arise when the *not odd(i)* and  $i < 0$  checks finally fail—it's time not to multiply.

*PowerN* also never computes anything not used in the final result. Failures (such as overflow) in intermediate computations occur only when failure is truly inescapable.

### Accuracy Considerations

To confirm that *PowerN* does indeed produce expected results, I have included a Turbo Pascal driver program called TPWRN.PAS (See Listing Two, page 84.)

Using TPWRN,  $7^{**i}$  is computed exactly and rapidly up to  $i = 14$ . Thereafter, the 39-bit effective precision (and 11-digit output rounding) are inadequate for confirming exact results with the CP/M-80 edition of Turbo Pascal.

On the other hand, comparative calculations for  $(1/7)^{**-i}$ , although mathematically the same as  $7^{**i}$ , deteriorate quickly. Value  $1/7$  cannot be carried exactly, and the discrepancy is quickly magnified in taking powers. By  $i = 14$ , the error exceeds 10, even though minimal error in  $7^{**14}$  has just shown up.

The final column of TPWRN output shows how much more quickly the calculus textbook approach, using  $\exp(i * \ln(7))$ , breaks down. Because this method is noticeably slower as well, there is nothing to commend it for exact-integer exponents.

In making use of these results, keep in mind that the *PowerN* vintage 1.xx algorithms are the best available in terms of providing a direct solution at minimum cost. Nevertheless, working with the same precision as the input data inevitably dooms you to some sort of error. It is wise to be mindful of the prospective errors, however insignificant you regard them to be.

### PowerN Method

*PowerN* makes rapid computation of integer powers using the fundamental method of Donald E. Knuth's algorithm 4.6.3A (see source 11). The number of floating-point multiplications is reduced by taking advantage of the laws of exponents, in form

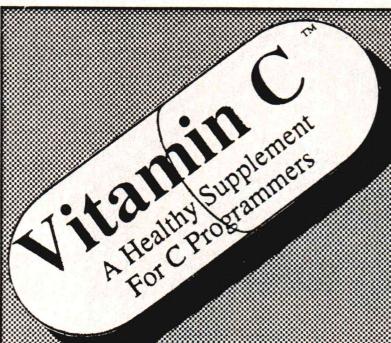
$$\begin{aligned} x^{**2i} &= \text{sqr}(x)^{**i}, & i > 0 \\ x^{**(2i+1)} &= x^{**2i} * x, & i > 0 \end{aligned}$$

In Knuth's formulation, as in *PowerN*, the first transformation is performed in an inner loop that doesn't stop until an odd exponent is inevitably reached. This idea is elegantly restated, at somewhat increased decision cost, by Dijkstra and Jensen and Wirth (see sources 3 and 10):

```
r := 1.0;
while i > 0
  do begin
    while not odd(i)
```

```
do begin
  x := sqr(x);
  i := i div 2
end;
r := r*x; i := i - 1;
end;
```

The lower decision cost of *PowerN* is obtained by not checking for even  $i$  quite so often. This applies a suggestion of David Gries to the effect that the inner *while* can be replaced by *repeat* upon arranging that  $i$  always be even whenever the  $i > 0$  test passes



### 100 % MONEY BACK GUARANTEE

Better than a brochure. More informative than a testimonial. Our guarantee gives you the opportunity to see first hand why programmers who demand performance are using Vitamin C.

**Find out for yourself why Vitamin C users are saying...**

*The best structured and documented C source package we have ever seen.*

*Thanks to Vitamin C, our projects are back on schedule.*

*I own them all, but I USE Vitamin C!*

**Vitamin C! \$149.95 + \$3 ground, \$6 second day air, or \$20 next day. TX add 6 1/8% tax.** Includes 100% source, reference manual, step by step tutorial, examples, sample programs. Specify Microsoft v3, Lattice, Aztec, Computer Innovations, DeSmet, or Mark Williams. Visa & MasterCard accepted. Ask about UNIX, TI-Pro, and other compatibility!

**For Orders Or More Information...**

**(214)243-6197**  
Creative Programming  
Consultants  
Box 112097  
Carrollton, Texas 75011-2097

### Data entry • Windowing Help System • More!

Finally! A library of high level C functions (not just a bunch of building blocks) designed to increase your productivity and help develop superior applications in dramatically less time! How? Well, Vitamin C automatically coordinates the complex tasks and leaves the programmer free to be creative! With Vitamin C, for example, you'll never even have to think about saving or restoring portions of the screen when a window is opened, closed or moved. Simply call wopen(), wclose() or wmove() and Vitamin C takes care of the complexities. It's just that easy! This philosophy of relieving the programmer from as many details as possible runs throughout Vitamin C. As a result, jobs that used to take days are finished in a matter of hours!

**Vitamin C's powerful features include...**

- Complete input formatting
- Unlimited validation
- Full attribute control
- Field sensitive help system
- Multiple virtual windows
- Fully automatic, collision proof overlay and restore
- Print to & scroll background windows
- Animated window "zoom"
- Move, grow, shrink, hide, or show any window
- Date & time arithmetic routines
- "Loop function" allows processing while awaiting input

*...and much much more!*

### ---PLUS---

All windowing & data entry features are already fully integrated for effortless data entry windows, display windows, and pop-up menus!

### Coming Soon... VCScreen!

Our new interactive screen "painter" actually lets you draw your input screens. Define fields, text, boxes & borders. Move them around. Change their attributes. Then after everything is just the way you want it, the touch of a button generates C source code calls to Vitamin C routines OR generates parameter files that will dynamically load & build each screen at run time. Either way, screen designs will be faster & more pleasing with VCScreen!

Circle no. 82 on reader service card.

## PASCAL

(Continued from page 37)

(see source 6). The saving of decisions is comparable to what Knuth obtains from the start using *goto*.

The main new idea in *PowerN* involves using the initialization (for guaranteeing that even powers always remain) to also replace the usual  $r := 1.0$  and first  $r := r^*x$  by a well-timed  $r := x$ .

The procedure is further optimized by using  $i \text{ shr } 1$  in place of  $i \text{ div}$

2 on Pascal implementations that permit it. Because  $i \text{ div } 2$  has the effect of also decrementing odd values of  $i$  in the one operation, separate adjustments to  $i$  are unnecessary.

$\text{PowerN}(x, 0) = x/x$  deserves special mention. It is not known what nonstandard assumptions are implemented in each computer arithmetic. If there is some provision for undefined operation  $0/0$ , however, you would wish to employ it. Use of  $x/x$  to catch  $x = 0$  or  $x = \text{nonstandard value}$  is then consistent with IEEE

proposals for floating-point arithmetic (see source 2). It is also as good as any other approach, assuming consistent extension of the laws of arithmetic and exponentiation to propagate indeterminate results:

$$x^{*0} = x^{*-0} = 1/(x^{*0}); \\ x/x = 1/(x/x);$$

or simply

$$x^{*0} = x^{*(1-1)} = x/x.$$

*PowerN* is contrived to propagate all cases without knowing in advance which ones actually apply.

## Availability

Current editions of *POWERN.PLB* (Listing One) and the *TPWRN.PAS* test driver are posted to the Borland International Forum (page BOR-100) on the CompuServe Information Service. Database DL1 of the forum provides ready-to-run Turbo Pascal versions.

I check onto CompuServe regularly. You can contact me via User ID 70100,271 for open discussions on either the CP/M Forum (go *CPMSIG*) or the Borland International Forum.

## Sources

Because powers are not as elementary as the standard operations of addition, subtraction, and multiplication, they (like general division and remainder operations) still lack widely recognized standard mathematical definitions. (Despite this condition, many programming language manuals and standards continue to omit definitions for their language's implementation of powers, long after the *ALGOL 60* report established the standard of precision I've tried to sustain here.)

This decade has seen renewed interest in computation of powers because the high-performance technique provides a lovely algorithmic method for generalization to several other problems. Even so, recent interest in describing computational methods for powers hasn't stabilized: Little slips are made in even the latest work, sometimes despite the existence of superior versions in older, accessible publications.

## C-terp

The C  
Interpreter  
You Won't  
Outgrow



C-terp will grow with you as you progress from novice through professional to guru. Unbelievable, but true, the easiest-to-use C interpreter will provide you with the most advanced programming features for upward growth. Our exclusive **object module support** enables you to add libraries (like HALO, PANEL, Windows for C, etc., or your own homebrew libraries) to C-terp as you add them to your computing repertoire. Use C-terp as a microscope on your libraries! Flip a bit and allow our **software paging** (NEW) to handle those big jobs! There are no fixed-size tables to overflow, and C-terp can be configured for different screens and screen adapters (NEW). With multiple modules and full **K&R support**, we offer a dream C environment.

- Our new improved **configurable editor** competes with anything going.
- Speed -- Linking and semi-compilation are breathtakingly fast.
- Convenience -- Errors direct you back to the editor with the cursor set to the trouble spot.
- Symbolic Debugging -- Set breakpoints, single-step, and directly execute C expressions.
- Compatibility guaranteed -- batch file to link in your compiler's entire library. Supported compilers include: **Computer Innovations C86, Lattice C, Microsoft C 3.0, Mark Williams C86, and Aztec C.**
- Many more features including batch mode and 8087 support.

### What Our Users/ Reviewers Are Saying

"...easy to use, powerful, and a timesaver."  
"...we absolutely LOVE C-terp."  
"...has restored my faith in interpreters."  
"...a programmer's dream."  
"...wonderful technical assistance."  
"...increased our productivity by a factor of 40."  
"...the best C product ever, in any category."

- **Price:** \$300.00 (Demo \$45.00)  
MC, VISA

**Prices include documentation and shipping within U.S. PA residents add 6% sales tax.**  
Specify compiler.

- C-terp runs on the IBM PC (or any BIOS compatible machine) under DOS 2.x and up with a suggested minimum of 256 Kb of memory. It can use all the memory available.

\* C-terp is a trademark of Gimpel Software.

## GIMPEL SOFTWARE

3207 Hogarth Lane • Collegeville, PA 19426  
(215) 584-4261

William. *Software Manual for the Elementary Functions*. Englewood Cliffs, N.J.: Prentice-Hall, 1980. If computational cost and exclusion of useful cases weren't enough of an indictment of  $\exp(n^* \ln(x))$  for  $x^{**} n$ , there are also accuracy problems to contend with. Chapter 7 describes the considerations that apply to sustaining maximum numerical precision in the computation of powers. *PowerN* doesn't escape such problems either, as demonstrated with the TPWRN driver.

2. Cody, W. J.; Coonen, J. T.; Gay, D. M.; Hanson, K.; Hough, D.; Kahan, W.; Karpluski, R.; Palmer, J.; Ris, F. N.; Stevenson, D. "A Proposed Radix- and Word-Length-Independent Standard for Floating-Point Arithmetic." *IEEE MICRO* 4, 4 (August 1984): 86–100. See the references, too. Articles in the special issue of *IEEE Computer* 14, 3 (March 1981) provide extensive discussion of the approach adopted for IEEE floating-point arithmetic standards.

3. Dijkstra, Edsger W. *A Discipline of Programming*. Englewood Cliffs, N.J.: Prentice-Hall, 1976. The sixth small example, pp. 65–67, provides some useful insight into the fundamental fast exponentiation technique, its underlying beauty, and the pitfalls of seemingly simpler alternatives.

4. Dromey, R. G. *How to Solve it by Computer*. London: Prentice-Hall International, 1982. Chapter 1, especially Sections 1.4–1.7, is enough to commend this book as a companion to a good Pascal manual and primer. All of Chapter 3 is relevant to the methods and application of *PowerN*, although algorithm 3.7 has the "late-termination inefficiency" of an extra  $\text{sqr}(x)$ , just as in sources 9 and 16. The careful discussion of the method, benefits, and special applications (Problem 3.7.3) make this treatment well worth attention, however.

5. Gries, David. *The Science of Programming*. New York: Springer-Verlag, 1981. A crisp derivation of the correctness of the fundamental transformation is found on pp. 239–240. It is an useful exercise to establish that the different form of the Knuth transform and its counterpart in *PowerN* preserve all the essential

conditions. (See sources 3 and 10 also.)

6. Gries, David. Personal communications. June 13, 1985 and August 14, 1985.

7. Hamming, Richard W. *Introduction to Applied Numerical Analysis*. New York: McGraw-Hill, 1971. Nowadays, more people than ever before seem to be using computation in dangerous ignorance of underlying assumptions, but the microcomputer is also a tool for demonstrating and comprehending the limitations of those very same methods. The first chapters of this book provide a good start on learning how not to surrender your fate to some printed formula.

8. Horowitz, Ellis, and Sahni, Sartaj. *Fundamentals of Computer Algorithms*. Potomac, Md.: Computer Science Press, 1978. Chapter 9 is of interest in conjunction with study of source 11, Section 4.6.4. Exercise 9–17 and the related discussion of Horner's method (pp. 424–428) make

unceremonious introduction of the Dijkstra-Jensen-Wirth version as a hint for evaluation of entire (sparse) polynomials. Additional ideas are offered in Exercise 9–22.

9. Hultquist, Paul F. "The Powers That Be." *PC Tech Journal* 3, 5 (May 1985): 213–214. Here's an alternative formulation to be compared against *PowerN*'s delegation of error handling to intrinsic system actions. Although this solution is also based on Knuth's algorithm 4.6.3A, its statement in Pascal has introduced a useless but overflow-prone squaring of  $x$  that Knuth avoided. (See source 11, exercise 4.6.3–1.)

10. Jensen, Kathleen, and Wirth, Niklaus. *Pascal User Manual and Report*. 2d ed. New York: Springer-Verlag, 1978. See Programs 4.8 (pp. 28–29) and 11.8 (p. 81). Jensen and Wirth recognize that if the exponent is even and nonzero, it must remain nonzero until after it is divided down to an odd value. This seems to be the

# C68 & C68/020

## C COMPILERS for

# MC680X0

- ▶ Produce highly optimized code
- ▶ Complete development environment: Assembler, Linking and Downline Loaders, Runtime Libraries
- ▶ Available for Motorola, DEC, and Alcyon host computers
- ▶ The #1 choice for compact, fast MC680X0 code
- ▶ \$1495 for C68 (Motorola host)  
\$2295 for C68/020 (Motorola host)



5010 Shoreham Place  
San Diego, CA 92122  
(619) 587-1155 TELEX 5106004947

DEC is a Trademark of Digital Equipment Corporation.

Circle no. 221 on reader service card.

# THINK AHEAD

Before you buy another  
Macintosh™ development system,  
consider two new  
compiled programming environments  
from THINK Technologies.

## Send me something to think about.

- I wish I had the missing link, send me information on Lightspeed™, the C compiler.
- I want to put the finger on bugs in my native code, send me information on Quicksilver™, the Pascal compiler.
- I'm a serious thinker, send me information on both.

NAME \_\_\_\_\_

TITLE \_\_\_\_\_

COMPANY \_\_\_\_\_

ADDRESS \_\_\_\_\_

CITY \_\_\_\_\_ STATE \_\_\_\_\_ ZIP \_\_\_\_\_

TELEPHONE \_\_\_\_\_

Send this coupon to:  
THINK Technologies  
420 Bedford Street  
Lexington, Massachusetts 02173  
Or call 617-863-5595

## PASCAL

(Continued from page 39)

earliest widely published example of using a nested *while* (*PowerN: repeat*) to deftly avoid questions for which the answer is already certain. Nevertheless, Section 11 (p. 160) gives a power function that always squares the base value once too often. This pattern is sustained in the latest edition as well.

11. Knuth, Donald E. *The Art of Computer Programming*. vol. 1, *Fundamental Algorithms*. 2d ed. Reading, Mass.: Addison-Wesley, 1973. Section 1.2.2 and Exercise 1.3.1–22 provide useful background on computation of powers.

Vol. 2, *Seminumerical Algorithms*. 2d ed. Reading, Mass.: Addison-Wesley, 1981. Section 4.6.3, Evaluation of Powers, also shows where further performance can be gained when there are special cases and global usage information to exploit. Section 4.6.4, Evaluation of Polynomials, can be even more valuable, though.

*PowerN*'s algorithm 4.6.3A heritage is now masked by the amount of reordering introduced to utilize Pascal control structures. Algorithm 4.6.3A is optimized around *odd(i)* or *i mod 2* being obtained as a byproduct of a single *i div 2* operation. This is particularly profitable when working in nonbinary representations. In contrast, *PowerN* is loaded with implicit assumptions about use of binary integers and the related efficiency of *odd(i)* and *i div 2* (or *i shr 1*) determination.

12. Knuth, D. E.; Lynch, W. C.; and Speroni, J. *Univac Solid-State Systems FORTRAN II Processor*. Univac Division of Sperry Rand, 1962. "X\*\*J Power-Routine" update, May 20, 1963, by Dennis E. Hamilton. This use in an equivalent of BCD arithmetic approached algorithm 4.6.3A simplicity by noticing that *i* must be odd when  $2*(i^5 \text{ dsh} 1) < > i$  and *dsh* is a decimal shift. It was also economical to detect shortcuts for  $x^{**2} = 1$  and  $r = 0$ . Similar tests were removed from *PowerN* because any benefit for microcomputer Pascal is doubtful compensation for the added overhead.

13. Newton, Isaac. "Letter of October

24, 1676". In *A Source Book in Mathematics, 1200–1800*. D. Struik, ed. Cambridge Mass: Harvard University Press, 1969. The common (pre-FORTRAN) notation for powers was introduced by Descartes and generalized by Newton and Leibniz [Cajori, Florian. *A History of Mathematical Notations*. vol. 1. Lasalle IL: Open Court, 1928. Sections 297–315].

14. Reynolds, John C. *The Craft of Programming*. London: Prentice-Hall International, 1981. Section 1.3.5 works carefully through several refinements just short of the Knuth formulation. In Section 4.2.5, the extraneous *sqr(x)* is avoided in a version that is rather tangled but systematically derived. There is, however, well-structured relief promised in a hint (Exercise 4.2.8–1) credited to David Gries.

15. Wirth, Niklaus. *Systematic Programming*. Englewood Cliffs, N.J.: Prentice-Hall, 1973. Chapters 1–9 provide just the sort of weaponry needed to derive modules such as *PowerN* (see Exercises 5.1–5.3) and also learn how to avoid needing *PowerN* in many cases (Chapter 9).

16. Wirth, Niklaus. *Programming in Modula-2*. 2d ed. New York: Springer-Verlag, 1983. An interesting variant on the *PowerN* method is discussed in Section 6.2 (pp. 20–23). Saving decrements in the second Power module is also at the cost of an extra squaring and possible overflow, however.

*PowerN* borrows the useful idea that *odd i* need not be decremented because  $(2^k + 1) \text{ div } 2 = (2^k) \text{ div } 2$  anyhow. It now takes more thought to properly state the second *while* condition and accompanying loop invariants, however.

DDJ

(Listings begin on page 84)

### Reader Ballot

Vote for your favorite feature/article.  
Circle Reader Service No. 3.

## IBM COMPATIBILITY at a not so IBM price



**TECH PC/AT . . . . . \$1999**

### PRICE INCLUDES:

- 6MHz 80286 CPU
- 512K
- One, 1.2 MB Floppy Drive
- 8 Expansion Slots
- 195 Watt Power Supply
- Complete MS DOS, PC DOS, Xenix Compatibility
- Runs Lotus 123, dBase III Framework and all other popular AT Software.
- ONE YEAR WARRANTY!!

### OPTIONS:

Tech PC/AT with 20MB Hard Disk . . . . . \$2399

Tech PC/AT with 20MB Hard Disk, Monochrome Monitor, Hercules® Compatible Mono/Graphics Card . . . . . \$2599

Also available with 6-8 MHz Switchable CPU, Tape Backups, Modems, Large Hard Disks, and Networking Systems.

**TECH TURBO PC/AT . . . . . \$2399**

6-8 MHz Switchable 80286 CPU

**TECH TURBO PC/XT . . . . . \$1099**

### PRICE INCLUDES:

- 4 to 7 MHz Software Switchable CPU
- 640K
- Two, 360K DS/DD Floppy Disk Drives
- 8 Expansion Slots
- 135 Watt Power Supply
- ONE YEAR WARRANTY!!

### OPTIONS:

Tech Turbo PC/XT with 20MB Hard Disk . . . . . \$1699

Tech Turbo PC/XT with 20MB Hard Disk, Monochrome Monitor and Hercules® Compatible Mono/Graphics Card . . . . . \$1899

**TECH PC/XT . . . . . \$799**

### PRICE INCLUDES:

- 4.77 MHz CPU
- 256K
- Two, 360K DS/DD Floppy Drives
- 8 Expansion Slots
- 135 Watt Power Supply
- ONE YEAR WARRANTY!!

### OPTIONS:

Tech PC/XT with 20MB Hard Disk . . . . . \$1399

Tech PC/XT with 20MB Hard Disk, Monochrome Monitor, Hercules® Compatible Mono/Graphics Card . . . . . \$1599

### TELEX: 272006

Answer Back-TECH

FAX: 714/556-8325

Visa, MasterCard, Check Accepted

**TECH PC PERSONAL COMPUTERS**

**714/754-1170**

2131 S. HATHAWAY, SANTA ANA, CA

92705

©1985 TECH PC  
\*Hercules is a registered trademark of Hercules Computer Technology.

\*IBM, IBM PC, XT, and AT are registered trademarks of International Business Machines Corp.

Circle no. 245 on reader service card.

# Learning Ada on a Micro

by Do-While Jones

***A Draw Poker program provides a vehicle for presenting some features of the Defense Department's favorite language.***

If you want to use the most technically advanced programming language available, you should learn Ada.

Validated Ada compilers are expensive and run on mainframe computers, but there is an affordable Ada subset that runs on CP/M-80 systems and that is complete enough to give you some useful experience. I wrote and tested all the examples in this article on a 62K CP/M system using a beta-test version of the Maranatha A compiler. A later version of this compiler is now sold under the name Supersoft A.

Ada's most attractive features show up only in large, complicated programs. Simple routines, such as sorting algorithms and finding factorials, don't demonstrate Ada well. That's why the following example is not an example of a program but is an example of a software development project.

Here's the problem. Imagine that you are starting a video-game business and that your first product is to be a Draw Poker game similar to those that are legal in Nevada.

You start off with this system requirement: The player puts as many coins in the game as he wants to bet, and it displays five playing cards. (Throughout this article, the feminine pronouns *she* and *her* always refer to Ada. The neuter pronouns *it* and *its* refer to an inanimate object, usually a computer or computer program. The masculine pronouns *he* and *his* refer to a male or female programmer or user.) The player decides to hold or discard each of the five cards. After the player has made his decision, the game displays the cards that replace those that the player has discarded

and evaluates the player's hand. If the player has a winning hand, it dispenses some coins determined by the player's bet and the value of his hand.

You are probably anxious to see the final result, so let's look at the solution in Listing One, page 86, before going through the step-by-step development.

### ***Easy to Read***

Listing One contains no comments, but because Ada is so easy to read, you might be able to understand the listing even if you have never seen an Ada program before. Ada is designed to be easy to read so that any programmer (not just the program author) can quickly (and correctly) modify the program. This reduces the software support cost—a major factor in software life-cycle costs.

The program begins by opening a new deck of cards called the *STOCK*. It then begins a loop that continues as long as the player wants to place a bet. It asks the player, "How many dollars do you want to bet?" and waits for him to enter a *WAGER*. The loop repeats until the *WAGER* is zero, and then the program ends.

If the player places a bet, the program shuffles the *STOCK* and deals five cards from it to the *PLAYERS\_HAND*. The *PLAYERS\_HAND* is displayed, and the player has an opportunity to discard as many cards as he

likes. The program then deals as many cards as are necessary to refill the *PLAYERS\_HAND* and displays the *PLAYERS\_HAND* again.

Next, the program computes the value of the *PLAYERS\_HAND*. If it contains a *ROYAL\_FLUSH*, the *PAYOUT* is 250 to 1. If the *PLAYERS\_HAND* has a *STRAIGHT\_FLUSH*, the *PAYOUT* is 50 to 1. The more common hands have lower returns, the lowest being *TWO\_PAIR*, which has a 2 to 1 *PAYOUT*. Any other hand (including a single pair) does not pay the player anything.

If the *PAYOUT* is 0, the program prints the message "Sorry, you lose." Otherwise, it tells the player what winning combination he has and tells him how much he has won. (The player's winnings are equal to the *WAGER* multiplied by the *PAYOUT*.)

After the program has told the player how much he has won, it goes back to the top of the loop and asks him how much he wants to bet this time.

The Draw Poker program illustrates the concept of top-down programming. The main program was written from the design specification without much regard for the details of how the program will actually do what it needs to do. At this point in the design, it is not necessary to know exactly how the program will get the player's bet, figure out if he has a winning hand, and drop the right number of silver dollars. The program must do these things somehow but the details come later.

Notice that the program reads very much like the system design specification does, which makes it easy to see if it addresses all the design requirements—you simply compare the program to its specification.

### ***An Object-Oriented Language***

Draw Poker was an easy program to

Another in a series of  
productivity notes on  
MS-DOS™ software  
from UniPress.

**Subject: Multi-window full screen  
editor.**

Multiple windows allow several files (or portions of the same file) to be edited simultaneously. Programmable through macros and the built-in compiled MLISP™ extension language.

**Features:**

- Famed Gosling version.
- Extensible through macros and the built-in compiled MLISP extension language.
- Dozens of source code MLISP functions; including C, Pascal, LISP and MLISP syntax checking.
- EDT and simple WordStar™ emulation modes.
- MS-DOS commands can be executed with output placed in an EMACS window.
- Run a compile on your program and EMACS will point to any errors for ease of debugging.

■ EMACS runs on the IBM-PC™ (XT/AT), TI-PC™, DEC Rainbow 100+, HP-150™ or any other MS-DOS machine. Requires at least 384K.

**Price:**

EMACS binary	\$325
EMACS source	995
One month trial	75
Also available for UNIX™ and VMS™	
Call for pricing.	

**UNIPRESS  
EMACS™**

**Subject: Powerful keyed file access,  
report generator and query  
language for MS-DOS.**

PHACT Isam is a keyed B+ tree file manager providing easy access to and manipulation of records in a database. PHACT-report produces formatted reports from PHACT databases. PHACT-query is a high level tool for manipulating data in PHACT databases.

**Features:**

- Supports fixed and variable length records (1-9999 bytes).
- Up to 9 alternate indices are supported.
- Record locking allows multiple simultaneous updates.
- Records can be accessed on full or partial key.
- Includes full Lattice™ linkable library and high-level functions.
- PHACT-report, an easy-to-use command language for formatting reports from existing PHACT databases.

■ PHACT-query can be used for making ad hoc and "canned" inquiries into a PHACT database.

**Price:**

PHACT ISAM	single-user	\$295
	multi-user	450
PHACT-report	single-user	165
	multi-user	275
PHACT-query	single-user	195
	multi-user	495
Source code available.		Call for terms.
Also available for UNIX and VMS.		Call for pricing.
		Call for pricing.

**ISAM FILE SYSTEM  
REPORT WRITER &  
QUERY LANGUAGE**

**PHACT™**

**Subject: Communication network for  
PCs.**

PCworks connects PCs to a variety of computers and information services.

**Features:**

- PCworks connects your PC to other PCs/Macintoshes™ or UNIX/VMS systems for file transfer and command serving.
- Menu driven for ease of use. Telephone numbers and other parameters can be stored.
- Transfer ASCII or binary files, send/receive electronic mail, or submit and print jobs in conjunction with UNIX or VAX™/VMS hosts.
- PC files can be printed on another system.
- Terminal Emulators include: ANSI, TTY, VT100/52.
- Protocol: error-free data transfers utilizing packet sequence numbers and checksums, or XMODEM protocol.
- Supports many modems/autodialers at a wide range of speeds.

**Price:**

PCworks	\$195
The Connectable Networks™	
Call for pricing.	

**PCworks™**

New from UniPress!

For our **Free Catalogue** and more information on these and other software products, call or write:  
UniPress Software, Inc.,  
2025 Lincoln Hwy., Edison, NJ 08817.

Telephone: (201) 985-8000.  
Order Desk: (800) 222-0550

(Outside NJ). Telex: 709418.  
European Distributor: Modulator SA,  
Switzerland Telephone: 41 31 59 22 22,  
Telex: 911859

OEM terms available.  
Mastercard/Visa accepted.

See our ad in the next issue for  
more MS-DOS products, including  
Lattice C compiler and Ps-Make.

SEE US AT UNIFORUM - BOOTH 1500

**UniPress Software**

Trademarks of UniPress EMACS and MLISP. UniPress Software, Inc.: WordStar, MicroPro Int'l. Corp., MS-DOS and IBM-PC, IBM, TI-PC, Texas Instruments; PHACT, PHACT Associates, Lattice, Lattice, Inc., UNIX, AT&T Bell Laboratories; VAX/VMS, DEC, Rainbow 100+, Digital Equipment Corp., HP-150, Hewlett-Packard Co., MACLINE, UniHost, PCworks and The Connectables Network, Touchstone Software, Corp., Macintosh, Apple Computer, Inc.

## LEARNING ADA

(Continued from page 42)

write in Ada because Ada is an object-oriented language. The first computer languages were equation-oriented—they were designed to solve problems that involved equations. Equations of motion, equations for solving polynomials, and equations for solving matrices are well known, but what is the equation of a draw poker game? It's true that people have written programs to play card

games in such equation-oriented languages as BASIC and FORTRAN, but some mental tricks were required to translate the game rules into equations and the cards into variables. Object-oriented languages let you solve the problem directly without those mental tricks—you can let your imagination run wild because you don't have to worry about practical details right away.

### A Visual Solution

Here's how I imagine a card game. I

see a deck of cards, called the stock. If I were imagining a bridge game, I would imagine two decks of cards, one with a red design on the back and the other with a blue design. In Ada I would express this concept as follows:

```
STOCK, RED_DECK,  
BLUE_DECK : Decks;
```

This indicates that STOCK, RED\_DECK, and BLUE\_DECK are three different objects, but they are all the same type of thing. They are all decks of cards, so I have defined them to have the type *Decks*.

Next, I remove the cellophane wrapper from a deck of cards and discard it (along with the jokers). I examine the cards and find they consist of 52 cards of 4 suits and 13 ranks, neatly sorted by suit and rank. In Ada, I can write a procedure to open a new deck of cards using the simple command

```
Open_New(STOCK);
```

The dealer picks up the stock and begins to shuffle it.

```
Shuffle(STOCK);
```

The dealer then begins dealing cards. He deals a card to you, one to me, and finally one to himself:

```
Deal_A_Card(YOUR_HAND);  
Deal_A_Card(MY_HAND);  
Deal_A_Card(DEALERS_HAND);
```

Again YOUR\_HAND, MY\_HAND, and DEALERS\_HAND are individual objects of the same type. In order to distinguish each of these objects from decks of cards, I define them to be of type *Hands*:

```
YOUR_HAND, MY_HAND,  
DEALERS_HAND : Hands;
```

*Open\_New*, *Shuffle*, and *Deal\_A\_Card* all show actions, so they act as verbs. STOCK, RED\_DECK, BLUE\_DECK, YOUR\_HAND, MY\_HAND, and DEALERS\_HAND are the objects of those verbs. I need to describe those objects to Ada before I can tell it how to do the actions the verbs require.

### Type Definitions

What is a deck of cards? It is a collec-



## FoxBASE. The DBMS That Bridges The Gap Between Single-user And Multi-user.

### Unsurpassed Program Development Speed.

FoxBASE™ uses a state-of-the-art B+ Tree index structure for quicker, more efficient data access. A sophisticated virtual storage technique becomes an invaluable timesaving device as it works to insure that frequently referenced programs are retained in memory in compiled form. What's more, FoxBASE provides automatic 8087/80287 math coprocessor support for ultraquick program execution speed—as much as six times the speed of dBASE II!™

### Highly Portable.

FoxBASE is much more than a relational database management system. Written in C, FoxBASE is an extremely portable interpreter/compiler. Now you can port from one machine or operating system to another without changing your applications. And this portability protects your investment in programs by insuring their use in future machine and operating system environments.

### dBASE II Compatible.

FoxBASE is both source language—including full macro usage—and data file compatible with the dBASE II database language. This means your existing dBASE II databases can be used unchanged. Furthermore, it puts thousands of public-domain and commercially available dBASE II programs at your disposal.

### Available Under Multi-user Systems.

Our multi-user versions of FoxBASE feature many additional enhancements. Like automatic file-locking and record-locking capabilities. Use of termcap, so FoxBASE can run on virtually any terminal. And, with some versions, a two billion record file capacity.

#### Multi-user Versions:

Xenix® \$995. MultiLink™ \$995.

IBM PC/XT™ \$995.

Single-user Versions:

MS/PC-DOS™ \$395. AOS/VS \$995.

**Don't be outfoxed by the others.  
Call or write Fox Software today.**

**FOXBASE™**  
From  
**FOX SOFTWARE, INC.**

27475 Holiday Lane, Perrysburg, OH 43551  
419-874-0162

Circle no. 94 on reader service card.

FoxBASE is a trademark of Fox Software, Inc. dBASE II is a registered trademark of Ashton Tate. Xenix is a registered trademark of Microsoft Corp. MultiLink is a trademark of The Software Inc. IBM PC/XT is a trademark of IBM Corp. MS/PC-DOS is a trademark of Microsoft Corp.

# Now You Know Why **BRIEF** is **BEST**

**"BRIEF, The Programmer's Editor, is simply the best text editor you can buy."**

John Dvorak, INFOWORLD 7/8/85

## The Program Editor with the **BEST** Features

Since its introduction, BRIEF has been sweeping programmers off their feet. Why? Because BRIEF offers the features **MOST ASKED FOR** by professional programmers. In fact, BRIEF has just about every feature you've ever seen or imagined, including the ability to configure windows, keyboard assignments, and commands to **YOUR** preference. One reviewer (David Irwin, DATA BASED ADVISOR) put it most aptly, "(BRIEF)... is quite simply the best code editor I have seen."

### **"A bona fide Undo..."**

Steve McMahon, BYTE 3/85

As Mark Edwards describes in DR. DOBB'S JOURNAL (11/85), "BRIEF has an outstanding undo facility. The default configuration allows the last 30 editing commands to be undone. This number can be raised to a maximum of 300 commands. Until you reach this maximum or run out of RAM, every command you issue can be undone. So if you make ten changes and then realize that the first one was an error, you can undo all the changes back to the mistake... Needless to emphasize, this facility can save endless grief."

*No other editor has this capability.*

### **Every Feature You Can Imagine**

Compare these features with your editor (or any other for that matter).

- FAST
- Full UNDO (N Times)
- Edit Multiple Large Files
- Compiler-specific support like auto indent, syntax check, compile within BRIEF, and template editing
- Exit to DOS inside BRIEF
- Uses all Available Memory
- Tutorial
- Repeat Keystroke Sequences
- 15 Minute Learning Time
- Windows (Tiled and Pop-up)
- Unlimited File Size -(even 2 Meg!)
- Reconfigurable Keyboard
- Context Sensitive Help
- Search for "regular expressions"
- Mnemonic Key Assignments
- Horizontal Scrolling
- Comprehensive Error Recovery
- A Complete Compiled Programmable and Readable Macro Language
- EGA and Large Display Support
- Adjustable line length up to 512

### **Program Editing YOUR Way**

A typical program editor requires you to adjust your style of programming to its particular requirements - NOT SO WITH BRIEF. You can easily customize BRIEF to your way of doing things, making it a natural extension of your mind. For example, you can create ANY command and assign it to ANY key - even basic function keys such as cursor-control keys or the return key.

### **The Experts Agree**

Reviewers at BYTE, INFOWORLD, DATA BASED ADVISOR, and DR. DOBB'S JOURNAL all came to the same conclusion - **BRIEF IS BEST!**

Further, of 20 top industry experts who were given BRIEF to test, 15 were so impressed they scrapped their existing editors!



### **MONEY-BACK GUARANTEE**

Try BRIEF (\$195) for 30 days - If not satisfied get a full refund.  
**TO ORDER CALL (800-821-2492)**

**Solution  
Systems™**

SOLUTION SYSTEMS, 335-D WASHINGTON ST., NORWELL, MA 02061, 617-659-1571

BRIEF is a trademark of UnderWare

## LEARNING ADA

(Continued from page 44)

tion of individual cards. In Ada, two data types—an array or a record—can represent a collection of things. An array is used to represent a group of items of the same type, while a record represents items of different types. Because of the fact that cards are all of the same type, it seems natural to use an array to represent a deck of cards:

type Decks is array(1 .. 52) of Cards;

When you try to use a deck of cards, you will find out that you need to know a little more about decks of cards, so this type definition will have to be improved, but let's let it stand as it is for a moment. It says that objects that are *Decks* have the structure of an array. This array has 52 elements, and each element is an object that has the type *Cards*. I must therefore define the data type *Cards*.

A card has two components—a suit and a rank. I can't use a two-element array because suits and ranks are not interchangeable, so they must be of different types. (The 52 elements in *Decks* are interchangeable, which allows me to shuffle a deck.) The data structure that holds a collection of different type elements is a record:

```
type Cards is
  record
    SUIT : Suits;
    RANK : Ranks;
  end record;
```

This says that an object of type *Cards* consists of a pair of objects, one object is called *SUIT*, and the other is called *RANK*. The object *SUIT* is of type *Suits*, and the *RANK* object is a *Ranks* type object.

Now I have to define *Suits* and *Ranks*:

```
type Suits is (CLUBS, DIAMONDS,
  HEARTS, SPADES);
type Ranks is (TWO, THREE, FOUR,
  FIVE, SIX, SEVEN, EIGHT, NINE, TEN,
  JACK, QUEEN, KING, ACE);
```

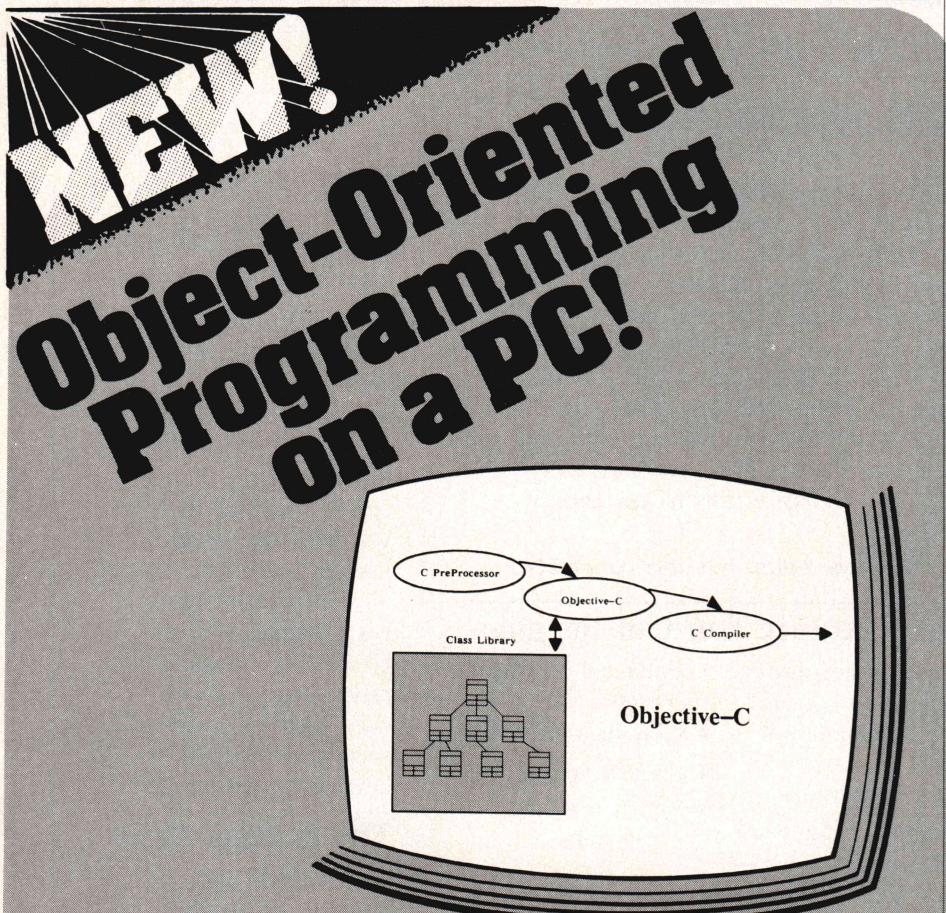
*Suits* is called an enumeration type because it has a finite number of values that can be enumerated. The four possible values are *CLUBS*, *DIAMONDS*, *HEARTS*, and *SPADES*. I could have listed them in any order if I were only going to play poker with this deck of cards, but I might some day want to play bridge. In bridge, clubs are the lowest ranking suit and spades are the highest. By listing them in the order above, I have given them their conventional order and can use *<* and *>* operators to compare the importance of two objects of type *Suits*.

Similarly, if I had said *type Ranks is (ACE, TWO, ...)*, then the *ACE* would have been the lowest ranking value. Defining a type by enumerating values not only defines what the permissible values are, but it also defines their order of relationship.

Finally, I define *Hands*.

```
type Hands is
  array(1 .. CARDS_IN_HAND)
  of Cards;
```

This definition looks a lot like the def-



Objective-C™ is an object-oriented programming language and a fully documented library of reusable components...adding messages, objects and inheritance to C language. Applications written in Objective-C are fully compatible with other Objective-C compilers running under UNIX, VMS or AOS.

Objective-C provides the productivity of object-oriented programming, while

retaining the portability and efficiency of C. PPI also provides comprehensive technology transfer to insure that your programmers fully understand this exciting new technology.

PPI's Objective-C compiler generates C code, which requires the Microsoft V3 C compiler running under MS/DOS.

At \$500 Objective-C is affordable. Order today!

PRODUCTIVITY PRODUCTS INTERNATIONAL  
27 Glen Road, Sandy Hook, CT 06482. (203) 426-1875.



Circle no. 140 on reader service card.

inition of a deck of cards, except I have used a named number to express the number of cards in a hand. The definition of a named number is:

```
CARDS_IN_HAND : constant := 5;
```

Poker hands have five cards. If I were playing bridge, I would substitute the number 13 for 5 in the definition of *CARDS\_IN\_HAND*, and then everywhere in the program, the correct number of cards would be used.

Maranatha A differs from Ada in that only the first ten characters of a name are significant. I therefore had to use *CARDS\_IN\_HAND* and *CARDS\_IN\_DECK* instead of *CARDS\_IN\_A\_HAND* and *CARDS\_IN\_A\_DECK* because Maranatha A would interpret both the latter as *CARDS\_IN\_A*.

When I started writing the *Shuffle* and *Deal\_A\_Card* procedures, I quickly discovered a problem with the types *Decks* and *Hands*. When dealing from the deck, I needed to know how many cards were left in the deck. When dealing cards to a hand to replace discards, I needed to know which cards had been played (that is, discarded). I therefore had to redefine those data types. This did not, however, affect any of the rest of the program that I had already written. (If I had been writing this in assembly language and had reserved only 52 bytes for each deck of cards and 5 bytes for each hand of cards, I would have had to change storage size and indexing schemes if I had changed the structure of *Decks* and *Hands*.)

The complete definition of a deck of cards includes the number of cards left in the deck and an array of cards:

```
type Decks is
  record
    CARDIS_LEFT : integer;
    FAN: Fans(1..CARDS_IN_DECK);
  end record;
```

I chose to call an array of cards a fan:

```
type Fans is
  array(integer range <>)
    of Cards;
```

This is a slightly more general array

definition than I have used before. The first array definition I used was *is array(1..52)*, which meant that the array index was an integer in the range 1 to 52. It may surprise you that I have to specify the index as an integer. Other languages may take this for granted because the index can't be anything other than an integer. Ada allows you to use any discrete data type as an index, so it is possible to write an array definition such as *is array(CLUBS.. SPADES)*. (You can't use a continuous data type

as an index because it doesn't make sense to look up the 3.14159th element of an array.)

The general form of the array definition uses the *< >* symbol, which is called a box in Ada. The general form allows you to define the array without forcing you to specify the size that it should be.

The complete definition of the hand of cards also has a fan of cards, but the fan in a poker hand has 5 elements instead of the 52 elements in a fan in a deck.

## Oh, Rapture!

### This is truly the editor I have been longing for.

- Dr. Joseph Newcomer

#### New Epsilon 3.0: fast, fully programmable text editor with an EMACS-style command set and concurrent processes!

Presenting Epsilon 3.0, the fastest, most powerful text editor available for personal computers. Epsilon has a built-in programming language, called EEL, for creating your own commands. Plus you get EEL source code for all of Epsilon's commands!

EEL has all the expressive power of the C programming language. It supports all C statements and expressions, pointers and user-defined structures. Unparalleled flexibility!

Because EEL looks like C, commands are easy to write. You don't have to learn a new language. Epsilon detects illegal pointer references, and has a source line single-stepping debugger and EEL profiler, too.

Our amazing Concurrent Process facility lets you run other programs while you continue to edit your files. The program's input and output are connected to a window, so you can edit them. Great for background compiles, debugging while looking at source code, and lots more!

Plus the advanced features programmers need:

- Concurrent Processes
- Multiple Windows
- Unlimited File Size
- Customizable Keyboard
- Tutorial
- Automatic Swap File
- Supports Large Displays
- Saves Deleted Text (n times)
- Context Sensitive Help
- Regular Expression Search
- Unlimited Number of Files
- File Name Completion
- Convenient Keyboard Macros
- Directory Perusal
- Language Support (C, Lisp, etc.)
- Uses All Available Memory

Epsilon runs on IBM PC's, XT's, AT's and compatibles with PC-DOS 2.0 or above and requires 256K of memory.

Epsilon is available directly from Lugaru Software Ltd, and costs only \$195.00. Order now using your Visa, MasterCard, or American Express card. Company purchase orders are also welcome. Order it today, so you can enjoy it soon!

Lugaru Software Ltd.  
5740 Darlington Road  
Pittsburgh, PA 15217  
(412) 421-5911

Circle no. 135 on reader service card.

## LEARNING ADA

(Continued from page 47)

```
type Hands is
  record
    PLAYED : Status(1..CARDS_IN_HAND);
    FAN : Fans(1..CARDS_IN_HAND);
  end record;

type Status is
  array(integer range <>)
    of boolean;
```

A hand of cards can be visualized as two five-element arrays. One array (the *FAN*) can be thought of as five slots that can each hold a single card. The second array (the *PLAYED* array) can be imagined as five signs placed in front of each card in the *FAN*. Each sign can have one of two messages written on it—it can say "This card has been played" or "This card has not been played."

Ada also has a predefined data type called Boolean, which can have

one of two values—*TRUE* or *FALSE*. This allows us to write simple expressions such as *if PLAYED = TRUE* then. . . . Some languages use 0 and -1 to represent true and false, so the programmer has to remember that 0 means a card has been played.

### Verb Definitions

Ada has two kinds of verbs—procedures and functions. So far, I have introduced three procedures (*Open\_New*, *Shuffle*, and *Deal\_A\_Card*) and no functions. I'll talk about functions later.

All procedures have the same general form as that given in Listing Two, page 86. There are five keywords (*procedure*, *is*, *begin*, *exception*, and *end*), followed by information specific to the procedure. For convenience I've called the locations that contain specific procedure information \*1 through \*5. The keyword *exception* and information in locations \*2, \*4, and \*5 are optional.

*Open\_New* in Listing Three, page 86, is a good example procedure because it uses all five keywords and has information in all five locations. *Open\_New* creates new decks of cards. The main program simply says *Open\_New(RED\_DECK);*, and the procedure creates a new deck of cards called *RED\_DECK*.

Notice that the statement ends with a semicolon. Some languages won't let a single program statement be more than one line long, but Ada ignores the carriage-return/line-feed sequence, so you can use them freely to make the program as readable as possible. It does mean, though, that you have to tell Ada where the statement ends by using a semicolon. (There is one exception—a carriage return marks the end of a comment.)

A procedure name, and possibly a formal parameter, go in the location marked \*1. Here *formal* is used in the sense of showing the form rather than meaning prim and proper.

In *Open\_New*, for example, *Open\_New* is the name of the procedure, and the object *DECK* is a formal parameter that stands for any object of type *Decks*. The word *out* tells Ada that *DECK* is an output from the *Open\_New* procedure and should not be erroneously used as an input. In other words, *Open\_New* considers *DECK* to be a write-only variable.

## Programmer Essentials

"Offers many capabilities for a reasonable price"

W. Hunt, PC Tech Journal

"I highly recommend the C UTILITY LIBRARY"

D. Deloria, The C Journal

### C ESSENTIALS

200 functions: video, strings, keyboard, directories, files, time/date and more. Source code is 95% C. Comprehensive manual with plenty of examples. Demo programs on diskette. Upgrade to THE C UTILITY LIBRARY for \$95.

\$100

### THE C UTILITY LIBRARY

Thousands in use world wide. 300 functions for serious software developers. The C ESSENTIALS plus "pop-up" windows, business graphics, data entry, DOS command and program execution, polled async communications, sound and more.

\$185

### ESSENTIAL GRAPHICS

Fast, powerful, and easy to use. Draw a pie or bar chart with one function. Animation (GET and PUT), filling (PAINT) and user definable patterns. IBM color, IBM EGA and Hercules supported (more soon). NO ROYALTIES. Save \$50 when purchased with above libraries. Available February, 1986.

\$250

**Compatible** with Microsoft Ver. 3, Lattice, Aztec, Mark Williams, CI-C86, DeSmet, and Wizard C Compilers. IBM PC/XT/AT and true compatibles.

**Compiler Packages:** Microsoft C - 319, Lattice or CI-C86 compilers -\$329. Save \$40 - \$50 when purchasing compiler and library combinations. Specify C compiler and version number when ordering. Add \$4 for UPS or \$7 for UPS 2-day. NJ residents add 6% sales tax. Visa, MC, Checks, PO's.

**ESI** ESSENTIAL SOFTWARE, INC  
P.O. Box 1003 Maplewood, NJ 07040 914/762-6605

Circle no. 138 on reader service card.

# Make your PC or AT into a COMMUNICATING WORKSTATION for only \$85

Use **ZAP**, the Communications System for Technical Users  
COMPLETE Communications for PROGRAMMING and ENGINEERING

EMULATION of graphics and smart terminals is combined with the ability to TRANSFER files reliably, CAPTURE interactive sessions, and transmit MESSAGES while also being able to swap between your mini or mainframe session and your PC application. SUSPEND a line to run a PC application. Reconfigure features to fit the communications parameters and keyboard requirements of the host computer software. Complete technical documentation helps you understand and fit ZAP to your style.

## HIGHLIGHTS OF ZAP:

- Emulate TEKtronix 4010/14 and DEC VT 100, 102, 52 including variable rows and columns, windows, full graphics, even half tones.
- Reliable *file transfer* to/from any mainframes and PCs including KERMIT and XMODEM protocols plus you get a full copy of KERMIT. Transfer speeds ranging from 50 to 38,400 BAUD. Session control include *printer dumps* and *save to disk*.
- **MACRO** and **Installation files** ("scripts") controllable by you.
- EMACS, EDT and VI "Script" files are included. ZAP is also used with other popular software including graphics products like DISSPLA and SAS/GRAPH.
- **CONFIGURABLE** to communications, terminal features on the "other end"; 1, 2 stop bits; 5, 6, 7 or 8 data bits; parity of odd, even, none, mark and space; remap all keys including the numeric pad and standard keyboard, set any "virtual" screen size.
- Full PC/MSDOS access to run any command or program that will fit in your systems memory. ZAP takes less than 64K.
- 9 Comm ports are supported by ZAP. Plus full color in text and graphics make use of the IBM color, EGA cards, or Hercules Monochrome.

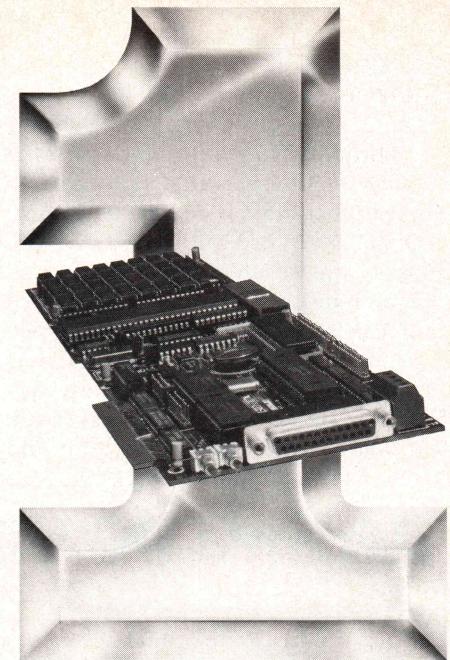
ONLY  
\$85

Full refund if not satisfied  
during first 30 days.

**Solution  
Systems**

335-D Washington St.  
Norwell, Mass. 02061  
617-659-1571  
800-821-2492

Circle no. 153 on reader service card.



## Number One in Performance

**68010/68000**

**Coprocessor for  
IBM/AT/XT/PC-**

**8/10/12.5mz No Wait States**

**\$1295<sup>00</sup> Cty. 1**

### FEATURES

- 1-2 MB RAM (1MB Standard)
- 16K-64K EPROM
- 2-8 Serial Ports
- Async/Sync/Bisync Communications
- Battery-backed Real Time Clock
- Battery-backed 2K-8K RAM
- 2 Parallel Ports
- 68881 Math Coprocessor
- Memory-mapped Dual-port BUS
- 3-9 Users Per Board (3 Standard)
- Up To 16 Boards Per AT/XT/PC
- Can Operate As Standalone Processor

### SOFTWARE

- OS9 (Powerful UNIX-like Multi-user OS)
- CPM/68K
- Software selectable OS including concurrent PC DOS/OS-9 or CPM/68K operation
- Support Module for IBM Graphics
- High-speed Local/Global Disk Caching
- Basic, Pascal, Fortran, C, and COBOL

IBM is a registered trademark of International Business Machines Corp. OS-9 is a registered trademark of Metrowerks Systems Inc. CPM/68K is a registered trademark of Digital Research Corp. MS-DOS and MS-DOS are registered trademarks of Microsoft Corp. UNIX is a registered trademark of AT&T



West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345  
East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450  
Distributor: Telemarketing Services, Inc.  
1897 Garden Ave., Eugene, OR 97403, 503/345-7395

Circle no. 174 on reader service card.

## Scrap your LINKER

with

# FASTER C

Reliably:

CUT Compile times (by 15% to 55%)

CUT Testing times (by 12% to 37%)

**HOW:** FASTER C keeps the Lattice C or C86 library and any other functions you choose in memory. It manages a jump table to replace the LINKER and immediately execute your functions. You can also CALL active functions interactively to speed your program debugging. It includes many options for configuration and control.

"Automatic" support for new libraries by reading the .OBJ files makes support for new libraries quick and simple.

AVAILABLE FOR PC-DOS, IBM-AT,  
AND ANY 256K MSDOS SYSTEM.

ONLY \$95.

Full Refund if not satisfied  
during first 30 days.  
Call 800-821-2492

**Solution  
Systems**

335-D Washington St., Norwell, Mass. 02061  
617-659-1571

Circle no. 148 on reader service card.

## LEARNING ADA

(Continued from page 48)

Location \*2 is used to create some temporary objects that will cease to exist when the procedure is finished. *Open\_New* creates two such objects—the variable *i* is an integer used as an index, and *CARD* is a blank piece of cardboard on which *Open\_New* prints a rank and a suit and then puts them in the *DECK*. You can give a variable an initial value when you create it. I gave *i* the initial value 0 but did

not give *CARD* an initial value.

The work gets done in the area marked with the third star. You've probably seen FOR . . . NEXT loops before but not ones like those in *Open\_New*. Usually FOR . . . NEXT loops are restricted to integers or, perhaps, real numbers. Ada can use any discrete variable as a loop index. The variable *S* is given the values *CLUBS*, then *DIAMONDS*, then *HEARTS*, and finally *SPADES*, as the outer loop is executed four times. Similarly, the variable *R* takes on the

values *TWO* through *ACE*.

Remember that objects of type *Cards* have two components—a *SUIT* of type *Suits* and a *RANK* of type *Ranks*. The compound name *CARD.SUIT* is "selected component" notation that means the *SUIT* object (or component) in the *CARD* object.

Each of the four times the program works through the outer (*Suits*) loop, it works through the inner (*Ranks*) loop 13 times. The index counter should be  $4 \times 13$  when all the looping is done, so the *CARDS\_LEFT* component of *DECK* should receive the value 52.

The last statement in the \*3 area is an error check. It is really a pair of statements as far as Ada is concerned. (Remember, a semicolon is an end-of-statement marker.) I like to put simple *if* statements, such as this one, all on one line because it is easier to read. Because Ada doesn't care about carriage returns, you can put multiple statements on one line if you wish.

The /= sign means not equal. Pascal programmers would probably prefer to use <> for that purpose, but the designers of Ada had already used that symbol for unconstrained arrays and wanted to have a separate symbol for inequality. It is an easy error for compilers to catch, so all the Ada compilers I have seen will tell you exactly what's wrong if you use <> by mistake.

The purpose of the *if* statement is to raise the exception *DECK\_ERROR* if the final value of the index (that is, the number of cards created) does not equal *CARDS\_IN\_DECK*. An exception is simply an error flag. If an error is detected, the statements in area \*4 are executed. Normally the exception handler is skipped, and program jumps down to the *end* statement. The *end* statement repeats the procedure name at \*5 so Ada can check to see if you accidentally left out an *end if* or *end loop* statement somewhere. The name in area \*5 must match the name in \*1 exactly, except that uppercase and lowercase differences are ignored. (*Open\_New* matches *Open\_new*, for example.)

# Atron's PC/AT Bugbusters

Hardware-assisted Software Debuggers for Bullet-proof PC/AT-based Products

### A BUGBUSTER STORY

Brad Crain, a project manager at Software Publishing (the people who developed both PFS:WRITE and PFS:FILE), relates the following: "On Friday, March 22, 1985, I was about to get on an airplane with Jeff Tucker, who was co-author of PFS:WRITE with me, and fly to IBM's Boca Raton, Florida facility. For a week, we had been unsuccessfully trying to isolate a bug in a new software product. In a last, desperation move, I set up an early-Saturday morning appointment with ATRON.

"Three of us walked through ATRON's door at 8:00 the next morning. Using ATRON's hardware-assisted debugging tools, we had the problem identified and fixed by 10:30AM."

Mr. Crain concludes: "We'd never have found the bug with mere

software debuggers, which have the bad habit of getting over-written by the very bugs they're trying to find. It doesn't surprise me that almost all the top-selling software packages were written by ATRON customers. Now that they've broadened their PC family of debuggers to include a PC/AT debugging tool, those of us seriously into 80286 development are greatly relieved."

### ARE YOU TRYING TO DO SOMETHING SCARY?

Like developing your AT-based software product in the dark? Without professional debugging tools?

Seven of the ten top-selling software packages listed by the *THE WALL STREET JOURNAL*\* were produced by ATRON customers. The PC PROBE™ bugbuster (\$1595) accounts for much of this success. Now that the PC/AT is the new standard for advanced commercial and scientific development, ATRON is proud to announce the AT PROBE™ bugbuster (\$2495). It has even more debugging capabilities than the PC Probe.



### HOW BUGBUSTERS KEEP YOU FROM GETTING SLIMED

The AT PROBE is a circuit board that plugs into your PC/AT. It has an umbilical which plugs into your 80287 socket and monitors all processor activity.

Since AT PROBE can trace program execution in real time, and display the last 2048 memory cycles, you can easily answer the questions: "How did I get here?" and "What are the interrupts doing?"

It can solve spooky debugging problems. Like finding where your program overwrites memory or I/O—impossible with software debuggers.

You can even do source-level debugging in your favorite language, like C, Pascal or assembler. And after your application is debugged, the AT PROBE's performance-measurement software can isolate your application's bottlenecks.

Finally, the AT PROBE has its own 1-MByte of memory. Hidden and write-protected. How else could you develop that really large program, where the symbol table would otherwise demand most of your PC/AT memory.

### BORLAND'S PHILIPPE KAHN: "THERE WOULDN'T BE A SIDEKICK™ WITHOUT ATRON'S DEBUGGERS."

So why waste more time reading through your program listing for the ten thousandth time, trying to find why your program starts howling with every full moon. Be like BORLAND, get your Atron bugbuster today and bust bugs tomorrow.



20665 Fourth Street • Saratoga, CA 95070 408/741-5900

\*WSJ, June 24, 1985, reporting Softsel figures. © 1985 by ATRON. PC PROBE™ and AT PROBE™ ATRON. SIDEKICK™ Borland. IBM Corp. owns numerous trademarks. Ad by TRBA.

### Packaging General-Purpose Routines

Listing Four is an Ada construct called a package. A package is a collection of data types and objects (con-

# Brand New From Peter Norton A PROGRAMMER'S EDITOR

only  
**\$50**

Direct from the  
man who gave you  
*The Norton Utilities*,  
*Inside the IBM PC*,  
and the *Peter Norton*  
Programmer's Guide.

## THE NORTON EDITOR



*Easily customized, and saved  
Split-screen editing  
A wonderful condensed/outline display  
Great for assembler, Pascal and C*

Peter Norton, 2210 Wilshire Blvd., #186  
Santa Monica, CA 90403, 213-826-8032  
Visa, MasterCard and phone orders welcome

Circle no. 243 on reader service card.

"This is the programmer's editor that I wished I'd had when I wrote my *Norton Utilities*. You can *program your way to glory* with *The Norton Editor*."

*Peter Norton*



## Number One In Performance

### Hard Disk Intelligent VCR Backup for AT/XT/PC

#### FEATURES

- High speed microprocessor controlled backup (68000)
- Two channel interface
- Built in LAN channel
- Software control of most VCR functions including Fast Forward, Rewind, and auto backup using VCR timer capabilities
- Economical VHS or Beta formats

## PCTEX Now for PC Users: Professional Typesetting Capability

PCTEX brings to the personal computer user the ability to put any kind of information on paper in a professional, elegant manner. It brings the full power and flexibility of TeX implementations on mainframes to owners of IBM PC's, AT's and workalike computers.

PCTEX is widely used for formatting technical and mathematical material. It is also perfectly suited for producing professional-quality reports, manuals, even books.

PCTEX offers a wide range of typefaces, and a wide choice of drivers which output the finished material on dot matrix printers (Epson, Toshiba), low-cost laser printers (Apple LaserWriter, Corona LP-300, HP Laser Jet) and graphics screen preview (Hercules, EGA). This ad was formatted by PCTEX and produced on a Corona LP-300.

Join hundreds of satisfied PCTEX users. Write or call us today.

PCTEX: only \$279. Dot-matrix drivers: \$100. Laser drivers: \$300. Preview (Hercules GC): \$250. MF Medley (44 fonts, including Computer Helvetica): \$100. Corona Laser Printer and PCTEX: complete \$3395. System requirements: DOS 2.0 or later, 512K RAM, 10M hard disk. M/C, Visa accepted.

Personal  
TeX  
Inc.

20 Sunnyside, Suite H  
Mill Valley, CA 94941  
(415) 388-8853 Telex 275611

Trademarks: PCTEX, Personal TeX, Inc.; TeX, American Mathematical Society; IBM PC and AT, IBM Corp; LaserWriter, Apple Computer, Inc.; Hercules Graphics Card, Hercules Computer Technology.

Circle no. 76 on reader service card.

**TLM Systems**

West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345  
East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450  
Distributor: Telemarketing Services, Inc.  
1897 Garden Ave., Eugene, OR 97403, 503/345-7395

Circle no. 175 on reader service card.

## LEARNING ADA

(Continued from page 50)

stants and variables) that are specific instances of those data types and operations that manipulate those objects. The package has a name (*PLAYING\_CARDS*) and can be used by other programs that ask for it by name. It is divided into two parts. The package specification (Listing Four A, page 86) explains what the objects and operations are but does

not give all the intricate details about the objects nor explain how the operations work. The second part (Listing Four B, page 88) is the package body, and it contains all those details.

There are several reasons for breaking the package in two. Most of them are because Ada was designed for large programs that require more than three programmers. Two or three programmers can usually work together easily, but when a program turns into a committee pro-

ject, it is harder for individual programmers to keep track of the whole program. Ada's package specification allows work to be partitioned and assigned logically. In this example, it tells one programmer that he must write a procedure called *Open\_New*, which creates an object of the type *Decks*, and it tells the second programmer that he can assume the existence of a procedure called *Open\_New* that will create decks of cards for him. The second programmer therefore does not need to wait for the first programmer to write *Open\_New* before he can begin his part of the job.

Another advantage of Ada's package specification is that if, for example, I discover a better way to shuffle cards, I can change the implementation of *Shuffle* in the package body without affecting any other part of the program. I can also be sure that no other programmer has made use of some special quirk in my previous method because no one knew how I did it before.

From a software vendor's point of view, the advantage in separating the body from the specification is that the body can be supplied in object code. The purchaser of the package then does not know how the package works, so he can't copy the design. If he wants to modify it, he has to pay the vendor to change it.

### Nested Packages

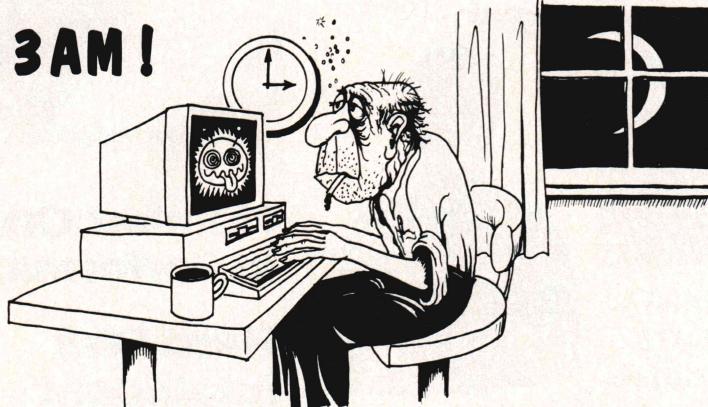
The first two statements in the package body of *PLAYING\_CARDS* (file *CARDB.ADA* in Listing Two) are:

with *CON\_IO*; use *CON\_IO*;

*CON\_IO* is a collection of console input and output routines. I could have used a common Ada package called *TEXT\_IO*, but I chose not to. Maranatha A doesn't check to see which package routines are needed, so it loads all of the *TEXT\_IO* package routines whether it needs them or not. I made a copy of *TEXT\_IO*, deleted all the disk interface routines, and renamed it *CON\_IO*, to reduce the size of the object code generated.

*TEXT\_IO* is not the only I/O package allowed in Ada, it is just one that is guaranteed to make your programs portable. (*TEXT\_IO* is the equivalent of CP/M BIOS—it provides a universal

It's 3 AM!



### Do you know where your bugs are?

This C programmer is finding his bugs the hard way...one at a time. That's why it's taking so long. But there's an easier way. Use

### PC-Lint

PC-Lint\* analyzes your C programs (one or many modules) and uncovers glitches, bugs, quirks, and inconsistencies. It will catch subtle errors before they catch you. PC-Lint resembles the Lint that runs on the UNIX\* O.S., but with more features and some awareness of the 8086 environment.

- Full K&R C
- Supports Multiple Modules—finds inconsistencies between declarations and use of functions and data across a set of modules comprising a program.
- Compares function arguments with the associated parameters and complains if there is a mismatch or too many or too few arguments.
- User-modifiable library description files for most major compilers.
- All warning and information messages may be turned on and off globally or locally (via command line and comments) so that messages can be tailored to your programming style.
- All command line information can be furnished indirectly via file(s) to automate testing.
- Use it to check existing programs, programs about to be exported or imported, as a preliminary to compilation, or prior to scaling up to a larger memory model.
- All one pass with an integrated pre-processor so it's very fast.
- Has numerous flags to support a wide variety of C's, memory models, and programming styles.
- **Price: \$139.00 MC, VISA**  
(Includes shipping and handling) PA residents add 6% sales tax. Outside USA add \$10.00
- Runs under MS-DOS\* 2.0 and up, with a minimum of 128Kb of memory. It will use all the memory available.

Trademarks: PC-Lint (Gimpel Software), UNIX AT&T, MS-DOS (Microsoft).

**NEW!!**  
Amiga - Lint  
Special Introductory Price  
\$98.00

### GIMPEL SOFTWARE

3207 Hogarth Lane • Collegeville, PA 19426  
(215) 584-4261

# Learn and Use AI Technology In Your First Evening With TransPROLOG

A complete *Prolog Interpreter, Tutorial, and set of Sample Programs:*

**Modify and write Expert Systems.**

Use the simple "Guess the animal" example on the Tutorial or use the sophisticated system for Section 318 of the US Tax Code written by one of the TransPROLOG authors and published in the March, 1985 issue of Dr. Dobb's Journal.

**Understand Natural Language**

Use the sample program that produces a dBase DISPLAY command as output.

**Programming experience is not required,** but a logical mind is. Serious development of experimental systems is practical with TransPROLOG. 1 or 2 pages in Prolog is often equivalent to 10 or 15 in C.

**RECENT IMPROVEMENTS:** MSDOS commands, on-line help, load Editor.

**AVAILABILITY:** All MSDOS, PCDOS systems.

**ONLY  
\$125**

Full refund if not  
satisfied during  
first 30 days.

**Solution  
Systems™**

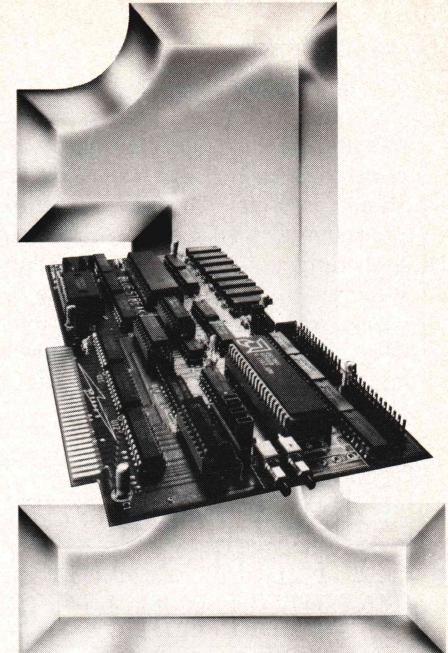
335-D Washington St.  
Norwell, Mass. 02061  
617-659-1571  
800-821-2492

Circle no. 152 on reader service card.

**Write Symbolic Math or Abstract Problem Solving Applications**

This is a complete Prolog program to convert from Farenheit to Centigrade: f\_to\_c(C,F):- C is(F-32) \*5/9. Planning programs and games are included to help you learn.

**BECOME FAMILIAR WITH PROLOG IN ONE EVENING.**



**Number One  
in Performance**

**Z80H  
BLUE STREAK™**

**IBM/AT/XT/PC- 8mz  
No Wait States**

**FEATURES**

- 64K-256K RAM
- 2K-8K EPROM/Static Ram
- 2 Serial Ports
- Async/Sync/Bisync Communications
- Real Time Clock
- Memory-mapped Dual-port BUS
- On-board/Remote Reset NMI capability
- Up To 32 Boards Per AT/XT/PC
- Can Operate As Standalone Processor
- Less Than Full Size Board  
(will fit other compatibles.)

**SOFTWARE**

- ZP/M tm CP/M Emulation Software  
(Supports Most CP/M Software)
- Multiuser Capability If Used As A Slave Processor

IBM is a registered trademark of International Business Machines.  
CP/M is a registered trademark of Digital Research Corp.



West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345  
East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450  
Distributor: Telemarketing Services, Inc.  
1897 Garden Ave., Eugene, OR 97403, 503/345-7395

Circle no. 173 on reader service card.

## LEARN LISP

Interactively and Write "Realistic" Programs  
with TransLISP for Only \$75

A "COMMON LISP" compatible Tutorial, Interpreter, Debugging, and Pretty Printer plus a Fast, Full Screen Editor, Samples and Help

**Start Easily and Quickly:**

A complete, modular tutorial helps you learn LISP at your own pace. An integrated, interactive environment provides all of the elements needed to enter, modify, analyze and debug programs.

**Write Realistic Programs:**

Short examples and substantial programs of about 10 pages in length help you learn by modifying, studying and using the key concepts needed to write programs of 1000 lines or more.

**The "COMMON LISP" Standard:**

TransLISP includes a 230+ function subset of the "COMMON LISP" Standard. Use extras like the MSDOS interface and graphics. Or use "strict compatibility" to make programs written in TransLISP, with no changes, work with other COMMON LISP systems like VAX LISP, GC/LISP or LISP Machine LISP.

Use and Modify the "Which Word Processor?" program  
to learn about EXPERT SYSTEMS.

**Runs on any MSDOS or PCDOS Systems:** Not copy-protected, TransLISP is available in just about any 3", 5" or 8" format. PC compatibles can run TransLISP with no installation procedure. 192K memory and 1 floppy drive are the minimums required.

**ONLY  
\$75**

For Beginners and Experienced Programmers

Full refund if not  
satisfied during  
first 30 days.

**Solution  
Systems™**

335-D Washington St.  
Norwell, Mass. 02061  
617-659-1571  
800-821-2492

Circle no. 155 on reader service card.

## LEARNING ADA

(Continued from page 52)

I/O interface to a specific hardware configuration.) I was not worried about portability because the final product will need special I/O routines that input and output silver dollars, which can't possibly be portable because of hardware limitations. *CON\_IO* is just a temporary I/O package that lets me substitute a CRT for a

coin detector and a dollar dispenser.

Maranatha A does not support generic routines, so *CON\_IO* includes an instantiated *INTEGER\_IO* package (rather than the generic package Ada would use).

The *with* clause tells Ada that a package named *CON\_IO* is stored in a file on the system disk. Ada should read this file to find out what data types, procedures, and functions have been defined by that package.

The *use* clause tells Ada that she can use these items as necessary to satisfy the program requirements.

*CON\_IO* contains a procedure to output a character string. The form of the procedure is:

```
put("A text string.");
```

Because Ada has been told to compile the Draw Poker program in the context of the *CON\_IO* package, she knows the definition of *put* so I do not need to define it.

## Mimicking Other Languages

You may have noticed that the package body of the *PLAYING\_CARDS* package includes the phrase *with APL; use APL;*. The computer language APL is different from most other languages because it has a function called *Deal* that is useful in programs that play cards. The *Deal* function returns a random sequence of numbers without repeating any number, which makes shuffling cards easy. Because I have already written a package to simulate some APL functions (the pertinent part is given in Listing Five, page 90), all I have to do is tell Ada that the *PLAYING\_CARDS* package needs to be compiled with APL, so all *APL* data types, procedures, and functions are available.

Unlike Maranatha A, true Ada would allow me to omit the phrase *use APL;*. Instead, I could use dot notation to tell it which package to use—that is, I could write a statement such as

SEQUENCE :=

```
APL.Deal(CARDS_IN_DECK,  
CARDS_IN_DECK);
```

This means that the variable *SEQUENCE* gets a value computed by a function in the *APL* package called *Deal*. The sequence (the first *CARDS\_IN\_DECK* in the argument list) contains 52 numbers in the range 1 to 52 (the second *CARDS\_IN\_DECK* in the argument list).

Ada knows where to find the *Deal* function, but a human reading the program might not. I wanted to use the dot notation to help remind me that *Deal* is an *APL* function. Maranatha A does not allow the dot notation, so I had to use a *use* clause instead.

# DISCOVER THE LANGUAGE OF ARTIFICIAL INTELLIGENCE PROLOG V

Interpreter for MS-DOS/PC-DOS

At last! A Prolog with enough muscle to handle real-world applications for UNDER \$100! Discover why Japan has chosen Prolog as the vehicle for their "Fifth Generation Machine" project to design intelligent computers.

CHOOSE FROM TWO GREAT VERSIONS:

**PROLOG V-Plus**

**\$99.95**

- More Than 100 Predefined Predicates
- Large Memory Model (to 640K)
- Floating Point Arithmetic
- 150-Page User's Manual and Tutorial plus Advanced Programming Documentation
- Co-Resident Program Editor
- Calls to Co-Resident Programs
- Text and Graphic Screen Manipulation

### STANDARD FEATURES ON BOTH:

- Clocksin & Mellish-Standard Edinburgh Syntax.
- Extensive Interactive Debugging Facilities
- Dynamic Memory Management (garbage collection)
- Custom-Designed Binder and Slipcase

### THE CHOICE OF UNIVERSITIES

Generous university site licenses and an excellent teaching tutorial and reference guide have made PROLOG V the choice of universities nationwide. Call for details.

PHONE ORDERS: 1-800-621-0852 EXT 468

<input type="checkbox"/> PAYMENT ENCLOSED \$	CA residents add 6% sales tax
<input type="checkbox"/> CHARGE MY:	<input type="checkbox"/> MasterCard <input type="checkbox"/> Visa
Card No. _____	Exp. Date _____
Signature _____	
Mr. / Mrs. / Ms. _____	(please print full name)
Address _____	
City / State / Zip _____	

PROLOG V-Plus \$99.95  
PROLOG V 69.95  
UPGRADE ONLY 40.00  
Return factory diskette and  
\$30 plus \$10 Handling

SHIPPING:  
\$ 5.00 U.S.  
7.50 Canada  
10.00 Caribbean,  
Hawaii Air  
20.00 Overseas Air  
COD Orders Not Accepted  
15 day check clearance



CHALCEDONY  
SOFTWARE

5580 LA JOLLA BLVD.  
SUITE 126 D  
LA JOLLA, CA  
92037  
(619) 483-8513

Circle no. 178 on reader service card.

## Nested Functions

In the Draw Poker program, I need to know if two (or more) cards have the same rank to see if I have two-, three-, or four-of-a-kind. If I couldn't nest functions (that is, if I could do only one function at a time), I would have to use several statements to find out if two cards have the same rank. For example, if I wanted to find out if cards 3 and 4 in *MY\_HAND* have the same rank, I would have to do something like

```
FIRST_CARD :=  
Card_Number(3, MY_HAND);  
FIRST_RANK :=  
Rank_of(FIRST_CARD);  
SECOND_CARD :=  
Card_Number(4, MY_HAND);  
SECOND_RANK :=  
Rank_of(SECOND_CARD);  
if FIRST_RANK =  
SECOND_RANK then ...
```

It is clearer to put it all in one statement, though:

```
if Rank_of(Card_Number(3,  
MY_HAND) = Rank_of  
(Card_Number(4, MY_HAND)  
then ...
```

The function *Card\_Number* pulls card 3 (or 4) out of *MY\_HAND*, and then the *Rank\_of* function reads the rank of that card.

## The Whole Program

The whole program is given in Listing Six, page 91. It begins with three comment lines that give the file name of the program, the date when it was written (or revised), and the author's name. The context clauses tell Ada that the *Draw\_Poker* procedure should be compiled in the context of the *CON\_IO* and *PLAYING\_CARDS* packages, so all the procedures, functions, and data types in those packages are defined and usable.

*Draw\_Poker* requires a special data type, called *Values*. The value of a poker hand can be *NOTHING* up to *ROYAL\_FLUSH*, so this type definition enumerates all the possible values in ascending order of importance.

*Draw\_Poker* also requires five variables (objects)—a *STOCK*; one *PLAYERS\_HAND*; a *WAGER* and a *PAYOUT*, which are both integers; and a

*VALUE*, which can have any of the values *NOTHING* through *ROYAL\_FLUSH*.

Next, the procedure *put* is defined for what seems to be the millionth time. Procedures that have more than one meaning are called overloaded procedures. *CON\_IO* has a *put* for individual characters, a *put* for character strings, a *put* for integers, a *put* for floating-point numbers, and a *put* for Boolean values. Listing Four contains a *put* for suits, a *put* for ranks, a *put* for cards, and a *put* for hands. Finally, I now have a *put* for values of poker hands.

If Ada wasn't smart enough to figure out which *put* to use, I would have to make up separate names to output integers, floating-point numbers, characters, strings, Booleans, suits, ranks, cards, hands, and values. Then I would have to remember which name I used for each data type. Because Ada allows me to reuse names, though, all I have to remember is that *put* outputs anything, and Ada has to remember how to output each object.

If an external package name is given (for example, *APL.Deal*), Ada will look only in that package for the procedure. If no external package name is given, she will look for a definition of the procedure in the declarative region (the area marked \*2 in Listing Two).

If Ada can't find the procedure in its list of procedures currently defined by the program, and there is a *use* clause, she will look for the procedure in the "used package." If there are two *used* packages and both have a procedure with the correct name and formal parameters, then Ada won't know which one to use. She will explain her dilemma during the compilation of the program (rather than just picking one at random), and you will have to provide an external package name to let her know which procedure to use.

**QUALITY SOFTWARE YOU CAN AFFORD!**

**Relocatable Z80 Macro Assembler**

- Only \$49.95 plus shipping.
- 8080 to Z80 Source Code Converter.
- Generates Microsoft compatible REL files or INTEL compatible hex files.
- Compatible with Digital Research macro assemblers MAC & RMAC.
- Generates Digital Research compatible SYM files.
- Conditional assembly.
- Phase/dephase.
- Cross-reference generation.
- Full Zilog mnemonics.
- INCLUDE and MACLIB FILES.
- Separate data, program, common, and absolute program spaces.
- Supports Hitachi HD64180.
- Z80 Linker and Library Manager for Microsoft compatible REL files available as an add-on to Assembler.

**ATTENTION Turbo Pascal Users:**  
Turbo Pascal Assembler will generate in-line machine code, include files.

**TO ORDER, CALL TOLL FREE:**  
1-800-367-5134, ext. 804  
For information or technical assistance:  
(808) 623-6361

Specify desired 5 1/4" or 8" format. Personal check, cashier's check, money order, VISA, MC, or COD welcomed.

**PRICE LIST**

Z80 Macro Assembler: \$49.95  
Assembler, Linker, and Library Manager: \$95.00  
Manual Only: \$15.00

Z80 Symbolic Debugger: \$49.95  
Manual Only: \$15.00

Assembler, Linker, Library Manager, and Debugger: \$134.95

Include \$5 for shipping and handling.

**MITEK**

Z80 is a trademark of Zilog, Inc. MAC and RMAC are trademarks of Digital Research, Inc. Turbo Pascal is a trademark of Borland International, Inc.

Circle no. 190 on reader service card.

### Separate Compilation

The *Draw\_Poker* procedure needs to compute the value of a hand. I chose to compile this separately, so I wrote a "body stub" where the function would normally go. The body stub is:

```
function Value_of(HAND : Hands)
  return Values is separate;
```

This tells Ada that a function called *Value\_of* operates on a *HAND* and returns an object of type *Values* (that is, *NOTHING* through *ROYAL\_FLUSH*) and that this function will be compiled separately. (See Listing Seven, page 91.)

Compiling *Value\_of* separately has three advantages. First, it divides the labor effectively. Sometimes adding more programmers to a project slows progress because each programmer has to know how another programmer has written part of the program before starting his own part. Separate compilation defines the interfaces between program modules clearly and allows all the programmers to work in parallel, thus shortening development time.

Second, separate compilation makes it easier to test procedures and functions. If the *Value\_of* function were an inseparable part of the *Draw\_Poker* procedure, think how long it would take you to find out if *Value\_of* recognizes when a hand contains a *ROYAL\_FLUSH*. Because *Value\_of* is compiled separately, though, you can write a program that tests *Value\_of* by asking what five cards are in the hand and then calling *Value\_of* to see if it figures out the value of the hand correctly.

Third, separate compilation makes it easy to fix mistakes. As a matter of fact, *Value\_of* does have a mistake. I don't play poker, so it wasn't until I showed the program in class that I found out aces can be high or low. *Value\_of* doesn't realize that *TWO, THREE, FOUR, FIVE, ACE* is really the same as *ONE, TWO, THREE, FOUR, FIVE* and should be considered to be a *STRAIGHT*. Because *Value\_of* is separate from *Draw\_Poker*, it was easily fixed, recompiled, and relinked to the other modules. (See Listing Eight, page 92, for corrections.)

Separate compilation and inclusion

of packages makes it difficult to compare Ada compiler speeds with other language compilers. It's fair to compare one Ada compiler to another on the basis of number of lines compiled per minute, but it isn't fair to compare Ada lines per minute to Pascal lines per minute (for example) because a Pascal compiler has to compile the whole program every time a change is made, while Ada just has to compile the part that has been changed. For a very long Ada program that has been properly divided into many modules, it might make more sense to measure the linker speed than the compiler speed.

### The Final Product

Remember that this case study involved making a video Draw Poker game that would be sold for profit. The program described here has no graphics, doesn't accept coins, and the user interface is through a terminal. The product, as it now stands, isn't commercially viable.

If I were going to market the product, I would get a mechanical engineer to design a coin detector that could reject slugs and count the number of genuine silver dollars entered. Meanwhile, I would write another procedure *get* so that *get(WAGER)* would accept input from the coin detector. I would put this procedure in a package called *HARDWARE\_IO*. I would then write a simple program based on this sequence of statements:

```
loop
  HARDWARE_IO.get(WAGER);
  CON_IO.put(WAGER);
end loop;
```

I could then drop coins in the slot and see if the CRT screen displayed the correct number of dollars.

The mechanical engineer would also design a dollar dispenser, which would require a *HARDWARE\_IO.put* routine. When the dollar dispenser and *put* procedure were finished, I could test them using a program containing these statements:

```
loop
  CON_IO.put("How much should I
              pay?");
  CON_IO.get(WINNINGS);
  HARDWARE_IO.put(WINNINGS);
end loop;
```

Simultaneously, an electrical engineer would work on a graphics interface that could display playing cards. It would need a *HARDWARE\_IO.put* procedure for *Cards*, which would replace the *PLAYING\_CARD.put* procedure.

### From Host to Target

I think you can see how Ada's modular nature makes it easy to put a system together. After you have tested each interface, you can integrate it into the Draw Poker program simply by replacing the *CON\_IO* procedure with the corresponding *HARDWARE\_IO* procedure.

Imagine how difficult it would be to get a working system if the Draw Poker program (including *HARDWARE\_IO*, *PLAYING\_CARDS*, *APL*, and *Value\_of*) were one long spaghetti-code program written by a team of programmers. Someone would always be waiting for someone else to finish something, and the program couldn't be tested until all the hardware was finished.

Ada's modular form lets you develop embedded system software on a host computer (even a micro) using simulated I/O drivers. You can develop test procedures using these simulated drivers, which you can then replace by real drivers in any order. You can test the real drivers quickly using the methods you used to test the simulated drivers. Integration should be a trivial task.

In the ideal case (that is, with unlimited manpower and resources), the time taken to complete the project is not much longer than the time taken to complete the longest task. The concept of a critical path (where one task cannot begin until another is finished) disappears.

### Micro Limitations

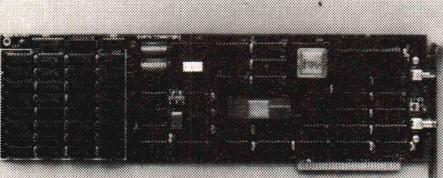
Maranatha A is not a true Ada compiler. It lacks overloaded operators, generics, and tasking, so you can't use it to develop programs that need these features. CP/M-80 systems typically have about 48K to 57K of usable program memory, so that limits the size of the program you can actually run on the micro host. Remember that Ada's modular form lets you compile and test modules separately, so you could test all the pieces of a huge program even if you couldn't

# LOOKING FOR AT PERFORMANCE FROM YOUR PC?



## EARTH HAS IT FOR LESS THAN \$1,000!

**YOUR SEARCH IS OVER!!** EARTH COMPUTERS' exciting new high-speed, 80286 accelerator card, **TurboACCEL-286™**, is just what you've been looking for. The **TurboACCEL-286** will boost your PC performance up to **Five times**...its completely software transparent...and its only \$995! **TurboACCEL-286** will function with most operating systems and application programs (unlike other so-called accelerator boards).



The **TurboACCEL-286** features a high-speed, 8MHz, 80286 processor, 512Kbytes of RAM (expandable to 1Mbytes), a switch for 8088 operation, and facilities for an 80287 math coprocessor. It occupies one expansion slot, is completely compatible with most PCs and is software transparent. End your search for AT performance. Order the **TurboACCEL-286** today! Call or write:



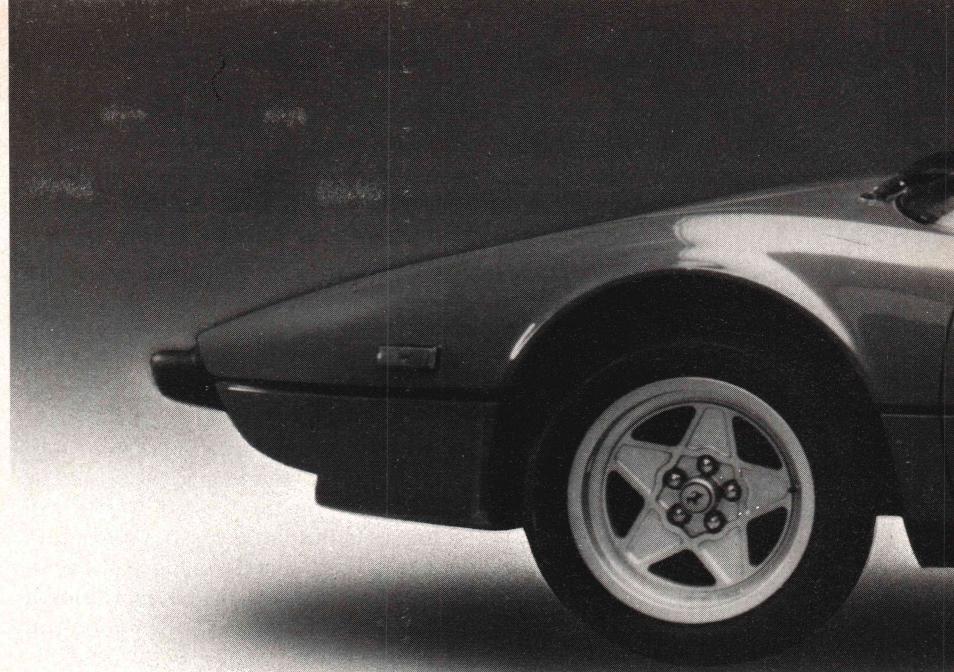
EARTH COMPUTERS

P.O. Box 8067, Fountain Valley, CA 92728  
TELEX: 910 997 6120 EARTH FV

**(714) 964-5784**

Ask about EARTH COMPUTERS' other  
fine PC and S-100 compatible products.

Circle no. 179 on reader service card.



## Finally, A Lint and Make for MS™ - DOS

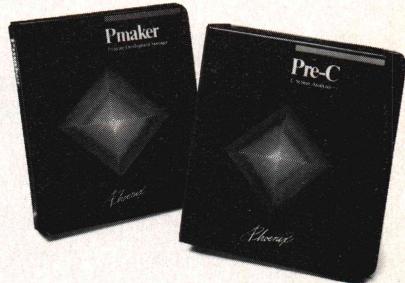
Get the full range of features C programmers working in UNIX™ have come to expect from their Lint and Make utilities. With Pre-C™ you can detect structural errors in C programs five times faster than you can with a debugger. Find usage errors almost impossible to detect with a compiler. Cross-check multiple source files and parameters passed to functions. Uncover interface bugs that are difficult to isolate. All in a single pass. Capabilities no C compiler, with or without program analyzing utilities, can offer. Pre-C outlives Lint, since you can handle analyses incrementally.

Pre-C's flexible library approach lets you maintain continuity across all programs in your shop, whether you use Pre-C's pre-built libraries, functions you already have, or some you might want to buy.

Plus, you're not limited to one particular library. Pre-C keeps track of all the libraries you're using to make sure that code correctly calls them.

With Pmaker™ you can update and track every module in your program. When you make a change in any source or include file, all you do

is run Pmaker. It will recompile changed modules and relink your program. With any compiler or linker you choose, Pmaker can update an object module library when one or several of the object modules are changed. You can use Pmaker to handle any task when a change requires several steps.



Pre-C by Phoenix. \$395. Pmaker by Phoenix. \$195.

Call (1) 800-344-7200.  
In Massachusetts (617) 762-5030.

Or, write: Phoenix Computer Products Corp., 320 Norwood Park South, Norwood, MA 02062.

*Phoenix*

PROGRAMMERS' PFANTASIES™ BY PHOENIX

Programmers' Pfantasies, Pre-C, and Pmaker are trademarks of Phoenix Computer Products Corporation. MS is a trademark of Microsoft Corporation. UNIX is a trademark of AT&T Bell Laboratories.

Circle no. 139 on reader service card.

## LEARNING ADA

(Continued from page 56)

run the whole program on the host.

I didn't feel hampered by Maranatha A. When I was beta-testing it, I didn't bother to read the whole *Maranatha A Language Reference Manual (LRM)*. I read just the Ada *LRM* and assumed Maranatha A would do what Ada does. I referred to the Maranatha manual only when Maranatha A wouldn't do something that Ada should do.

Maranatha A did not then have overloaded operators (it may have them by the time you read this), so I couldn't use the plus sign to add cards to a hand. I would have liked to have been able to write

```
PLAYERS_HAND :=  
PLAYERS_HAND  
+ Top_of(STOCK);
```

but as I couldn't, I used

```
Deal_A_Card(PLAYERS_HAND,  
STOCK);
```

instead.

I wanted to use dot notation to emphasize that the *Deal* function is in the *APL* package, but I suppose that I could have just as easily used a comment to do that.

Maranatha A doesn't have an IMAGE attribute, which made the *put* procedures for *Suits* and *Ranks* more complicated than they need be. For enumeration types with 4 to 13 values, it isn't too bad to have to use a case statement as I did, but if the enumeration type had 50 or 100 values, it could be annoying to have to output values in that way.

The extraneous semicolon in *separate (Draw\_Poker);* confused me for a while, but I think that error has been removed from the version that Supersoft is now selling. The exception handler also contained a bug, which forced me to include the

guard in the expression:

```
when DECK_ERROR!  
CONSTRANT_ERROR =>  
raise DECK_ERROR;
```

If not for the bug I could simply have written *raise DECK\_ERROR;* I think that bug has also been fixed in Supersoft's version.

It is a minor inconvenience that only the first ten characters of a name are significant. I lived for years with a FORTRAN language that permitted only six-character names, and it was picky about what the first letter was.

Now that I have a real Ada compiler with generics and tasking, I realize how valuable they are, but when I wrote the Draw Poker program I didn't know what I was missing, so it didn't bother me much. The Draw Poker program didn't need them, anyway.

The worst problem with my beta-test version is that a missing semicolon causes the compiler to crash. This was a real nuisance at first because I did not know any Pascal before I learned Ada, so I wasn't in the habit of ending statements with semicolons, and it crashed often.

The built-in random-number generator is not standard with Ada, and it was a handy function to have for Draw Poker. If I ever want to run this program on a validated Ada compiler, I will have to write a random-number generator to replace *RND* in the *APL* package.

Using a micro to develop Ada programs isn't ideal. It would be better to use a validated Ada compiler on a mainframe computer, but if you don't have \$30,000 to spend on a compiler (and a VAX-780 to run it on), then Maranatha A is a good alternative.

## MS-DOS, UNIX, APPLE MAC, CP/M, NETWORKS and MORE. ONE c-tree ISAM DOES THEM ALL!

The creator of Access Manager™ brings you the most powerful C source code, B+ Tree file handler: **c-tree**

- multi-key ISAM and low-level B+ Tree routines
- complete C source code written to K&R standards
- single-user, network and multi-tasking capabilities
- fixed and variable record length data files
- virtually opened files accommodate limited file descriptors
- no royalties on application programs

## \$395 COMPLETE

Specify diskette format:

- 5 1/4" MS-DOS
- 8" CP/M
- 3 1/2" Mac
- 8" RT-11



For VISA, MC and COD orders  
call (314) 445-6833

FairCom  
2606 Johnson Drive  
Columbia, MO 65203

© 1985 FairCom

The following are trademarks: c-tree and the circular disk logo—FairCom; MS—Microsoft Inc.; CP/M and Access Manager—Digital Research Inc.; Unix—AT&T; Apple—Apple Computer Co.

Circle no. 93 on reader service card.

DDJ

(Listings begin on page 76)

### Reader Ballot

Vote for your favorite feature/article.  
Circle Reader Service No. 4.

# NEW LANGUAGE BREAKS OLD RULES. GIVES PROGRAMMERS POWER, SPEED AND SIMPLICITY.

Try this remarkable language, PROMAL™, for 30 Days AT NO RISK and...

We think you'll be thrilled with this breakthrough system when you discover its power, ease of use, and dazzling performance on your IBM PC, Apple IIe/IIc, or Commodore 64. **But we don't expect you to accept our claims for PROMAL without proof**, so we invite you to explore the power of PROMAL on your own during our 30-day trial period.

## Broken Rules

Now that PROMAL 2.0 has broken the rules, a structured language doesn't have to be slow, unwieldy and difficult to use. PROMAL is fast, elegant, and simple.

## What Is PROMAL?

PROMAL stands for PROgrammer's Micro Application Language. But PROMAL is more than a high-level language, it's a total structured programming development system with a fast, one-pass compiler, a versatile full-screen editor, plus an integrated machine-language subroutine library. And for APPLE and Commodore systems it includes a DOS-like system "Executive."

## Better By Design

PROMAL was designed from "scratch" for **optimum performance** and ease of use on microcomputers. It has a simplified syntax with no awkward terminators

## PROMAL 2.0 FEATURES

### COMPILED LANGUAGE

- Structured indentation syntax
- No line numbers or terminators
- Long variable names (31 characters)
- Global, Local, & Argument variables
- Byte, Word, Integer & Real data types
- Decimal or Hex number types
- Functions & Procedures with passed arguments
- Predefined DATA of any type
- Multi-Dimensional Arrays (any type)
- Strings & pointers
- Control Statements: IF-ELSE, WHILE, FOR, CHOOSE, REPEAT-UNTIL, BREAK, NEXT, INCLUDE, ESCAPE, REFUGE
- Bit-operators, shifts, type casts
- Variables at any memory location
- Simple Machine Language interface
- Recursion supported
- Program chaining and overlays (IMPORT/EXPORT)
- Separate compilation of modules
- Load and run relocatable M/L programs
- Compile errors trapped for Editor

### EXECUTIVE (APPLE II & C64 Only)

- Command driven, with line editing
- Multiple user programs in memory at once
- Function key definitions
- Program abort and pause
- Prior command recall
- I/O Re-direct & batch jobs
- "DOS"-like commands: COPY, RENAME, DELETE, display FILES, TYPE, HELP, etc.
- Memory MAP, SET, and display commands

### EDITOR

- Full-screen, cursor driven
- Function key controlled
- Line insert, delete, search
- String search and replace
- Block copy, move, delete & file read/write operations
- Auto indent, indent support

### LIBRARY

- 50 Resident Machine-language commands
- Call by name with arguments
- String handling (9 routines)
- Re-directable I/O (STDIN & STDOUT)
- Formatted numeric output
- Decimal & Hexadecimal I/O
- Block fill/move/read/write
- Cursor control & line editing
- Data type conversion
- Random number function
- Real function support (in PROMAL): ABS, ATAN, COS, EXP, LOG, LOG10, POWER, SIN, SQRT, TAN
- Modem device support & much more

like ";" or "}" and **indentation is part of the syntax**, so structuring your code is natural and easy. Just compare PROMAL with BASIC in this example:

### Equivalent Program Segments

<b>PROMAL</b>	<b>BASIC</b>
REPEAT	11910 REM
PROMPT_AT 5,24, "Add Chg/Quit?"	11920 CL = 5,LN = 24,PRS = "Add Chg/Quit?"
IF Reply = 'A'	11925 GOSUB 9490:REM GET REPLY
ADD Item	11930 IF CL = 5,LN = 24,PRS = "Add Chg/Quit?"
New_Items = New_Items + 1	11940 I9 = IT:GOSUB 10100:REM ADD
ELSE IF Reply = 'C'	11945 NI = NI + 1:GOTO 11920
CHANGE Item	11950 IF PRS <> "C" THEN 11970
UNTIL Reply = 'Q'	11960 I9 = IT:GOSUB 6050:REM CHG
	11970 IF PRS <> "Q" THEN 11920

PROMAL is readable and understandable. You see the logic from the structure. And PROMAL lets you call procedures by name – so no more GOSUBs. But there's more.

## Slick Editor

Editing your source is a snap with the specially-designed and integrated **full-screen Editor** – it not only helps you structure your program, it **even finds compilation errors** – automatically.

## Quick Compiler

The compiler is a **lightning-fast**, one-pass, recursive descent design. On the IBM PC it crunches source to object at **2000 lines per minute**, and it's equally impressive on the Apple and C64. And your PROMAL source code is **portable from machine to machine**. That means your source can be used on all PROMAL target machines.

## Run-Time Speed Demon

PROMAL blows away Apple II and C64 languages from BASIC and PASCAL to FORTH. (Send \$3 for a copy of our full benchmark report.) It's **2000% faster** than BASIC. And on a normal IBM PC, the native 8088 code from **PROMAL beat Turbo Pascal 3.0** by 10% on the standard sieve benchmark!

## DOS For Those Without

If you don't have a real "DOS," then PROMAL gives you a **true operating system** environment with the built-in operating system Executive. (See box.)

### Order Form for PROMAL 30-Day Trial!

My system is (check one):

IBM PC/100% compatibles    APPLE IIc/IIe  
 COMMODORE 64/128

Please RUSH me:

PROMAL Developer's System – Compiler, Editor, Library, Demo disk, 280-page manual, (Plus Executive for Apple & C-64) and stand-alone program generation (no royalties).

**\$99.95 + 5.00 Shipping & Handling**

End-User System for Apple IIc/IIe and Commodore 64/128 – all features of Developer's Version except stand-alone program generation (Executive needed for program execution).

**\$49.95 + 5.00 Shipping & Handling**

Graphics ToolBox (Apple/C64 only) – 20 routines for hi-res graphics: windows, clipping, text-on-graphics using scaled, rotated, user-defined fonts.

**\$29.95 + 2.50 Shipping & Handling**

Please charge my

Visa  
 MasterCard  
 American Express  
 My check is enclosed

Card Number \_\_\_\_\_

Signature \_\_\_\_\_ Exp. Date \_\_\_\_\_

Name \_\_\_\_\_

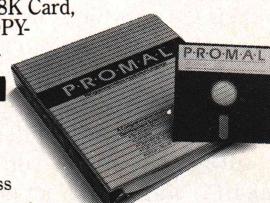
Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_

NC residents add 4 1/2% sales tax.

Foreign orders add \$20.00 additional shipping.



PROMAL is a trademark of Systems Management Associates, Inc. Turbo Pascal is a trademark of Borland International, Inc.

Circle no. 104 on reader service card.

## Outside Opinion

Naturally we're enthusiastic about PROMAL, but here's what other programmers are saying:

"Excellent... an ideal development system. . . . Well done indeed!"

M. T. V.  
Naperville, Ill.

"I am... so amazed by PROMAL... I cannot believe the high degree of excellence of this entire package."

C. P., Ph.D.  
Ridgeway, New York

"I don't know that I've ever seen a [system] as thoughtfully designed and as skillfully executed as PROMAL. Its logic and ease of programming are truly remarkable. Its speed of execution is phenomenal... congratulations."

E. C. R.  
Alexandria, VA

## Safety In Numbers

SMA, Inc. has been satisfying customers (over 100,000) since 1982 with innovative microcomputer products. Now you can join our thousands of satisfied PROMAL users, by trying it today.

## Try It For 30 Days On Us

Send us some bucks and we'll send you PROMAL on trial for 30 days. If for any reason whatsoever you are not satisfied, just send it back for a quick refund of your purchase price. No questions asked. No risk.

## How To Order.

Call TOLL-FREE to order with your credit card or use the handy order form below to send in your check or money-order for your 30-day trial. Don't wait, you deserve the power of PROMAL today!

**1-800-762-7874**

In NC: 919-878-3600

 Systems Management Associates, Inc.  
3325 Executive Drive, Dept. D-2  
Raleigh, North Carolina 27609

PROMAL runs on IBM PC/PCjr with 192K, Commodore 64/128, APPLE IIc, or APPLE IIe with 80 Col. 128K Card, and is NOT COPY-PROTECTED.

# Overview of the DOD Ada Software Repository

by Richard Conn

***With Ada becoming available for micros, this base of public-domain software is a good way to learn. The Ada Software Repository is over 20 megabytes in size.***

A repository of Ada programs, Ada software components, educational material, and Ada-oriented information has been established on the SIMTEL20 host computer on the Defense Data Network (DDN). This repository, established on November 26, 1984, is accessible to thousands of host computers on the more than 80 subnetworks that comprise the DDN. Access to this repository will be provided to the Ada community at large in the near future; with validated Ada compilers becoming available for popular microcomputers, this large base of public-domain Ada software and information will be a good way to introduce yourself to Ada and learn more about it.

The repository on SIMTEL20 serves two purposes: to promote the exchange and use of Ada programs and tools (including reusable Ada software components) and to promote Ada education by providing several working examples of programs in source form for people to study and modify. It also contains other useful information.

The repository is divided into several subdirectories by topic. Table 1, page 61, shows the general topic areas, the subdirectory names, and the sizes of the documentation and source-code files in each subdirectory. Today the DOD Ada Software Repository contains more 300 files, totaling more than 20 megabytes in size.

## ***Tour of the Ada Repository***

In the following paragraphs, I will take you on a brief tour of the Ada Software Repository. I will cover only selected points of interest and follow Table 1.

The first topic is "General Information," under which is the Education subdirectory. A glossary of Ada terms, a listing of Ada and Ada-related textbooks, a bibliography of Ada textbooks (which includes comments on the books and their audiences), an example of object-oriented design, productivity data (which presents data from live projects coded in Ada), and an example of how Ada can interface with other languages are some of the items of information available here.

The General subdirectory contains many tidbits of information, ranging from databases on the repository to tutorials on how to transfer files on the DDN.

The Pointers subdirectory contains information on where to look for sources of information and Ada software outside the repository. The

large collection of Ada compiler benchmarks on USC's ECLB host computer, the address of the DOD Software Engineering Institute, the INFO-ADA database on ECLB (which provides listings of conferences and current events), and a list of all validated Ada compilers are some of the items contained or referenced in this subdirectory.

The second topic is "Reusable Components." Reusability of software without the need for a complete redesign is supported by Ada, and subdirectories such as Components, Math, and Virterm were established to contain software components that may be reused time and again in various applications.

Three dynamic string packages, a generic quick-sort, two linked-list packages, a Unix-style ARGC/ARGV parser, and a command-line interpreter are some of the items found in Components. Math includes a math library of log, trig, exponential, and other functions and libraries of matrix-, bit-, and set-manipulation routines. Virterm contains virtual terminal packages, including a Curses package modeled after the Curses of Berkeley Unix.

The third topic, "Software Development Aids," includes toolsets for compilation order, Ada style, and metrics analysis of Ada programs; software project-management tools; Ada cross-reference programs; three Ada pretty printers; and a library file manipulation tool (Pager, which groups smaller files into larger library files).

Richard Conn, 6300 Roundrock, Apt. 3008, Plano, TX 75023

The communications software includes an implementation of the TCP/IP communications protocol (the DOD standard) and file transfer (FTP), mail handling (SMTP), and host-to-host (TELNET) communications programs.

For graphics fans, the Graphical Kernel System (GKS) has been implemented in almost 2 megabytes of Ada code.

Finally, a library of Ada compiler benchmarks (different from the one on ECLB), two text-file editors, a spelling checker/corrector, an LALR(1) grammar for Ada that may be processed under the LEX/YACC tools of Unix, an EMACS front-end for the development of Ada software, and a program that can configure a TVI970 terminal are some of the items found under the "Other Topics" category.

### How to Get Software

If you access a computer on the DDN (on the ARPANET, MILNET, or any other DDN subnet), you may subscribe to the Ada Software Repository mailing list by sending an electronic mail message to `ADA-SW-REQUEST@SIMTEL20`. In response to your message, you will be placed on the ADA-SW mailing list, over which notices of changes, updates, bugs, and other repository-oriented information are distributed. You will also receive a welcome message that details how to access the repository from a DDN host computer.

If you do not have access to a DDN host computer, a nine-track magtape distribution facility is being established from which any member of the Ada community will be able to acquire a tape of the entire Ada Software Repository. Also, submission of some or all of the software and documentation in the repository to the public-domain PC-BLUE library is being considered. */DDJ will announce these public-access opportunities when they become official.—ed/*

Limited DDN access is available to members of the Ada community with approval of the Ada Information Clearinghouse. A public account sponsored by the AdaIC on a DDN host computer at the University of Southern California (USC-ECLB) host computer provides access to current information on the Ada initiative and the ability to link with the SIMTEL20 host computer, scan the subdirec-

ties of the DOD Ada Software Repository, and transfer files from the repository. These files can then be viewed on the ECLB computer or transferred to a PC. To obtain the telephone number and access codes required to access the ECLB ADA- INFORMATION account maintained by the AdaIC, send conventional mail or telephone:

Ada Information Clearinghouse  
IIT Research Institute  
Ada Joint Program Office  
Room 3D139 (400 AN)  
The Pentagon  
Washington, DC 20301  
(703) 685-1477

The DOD Ada Software Repository is supported by the U.S. Department of Defense, U.S. Army, White Sands Missile Range.

DDJ

**(Listings begin on page 86)**

### Reader Ballot

Vote for your favorite feature/article.  
Circle Reader Service No. 5.

Topic	Directory	Documentation (Bytes)	Source Code (Bytes)
General Information	EDUCATION GENERAL NOSC-TOOLS POINTERS	195,010 515,205 121,531 133,680	0 0 0 0
		965,426	0
Reusable Components	COMPONENTS MATH VIRTERM	96,831 12,367 913,414	859,493 138,761 597,130
		1,022,612	1,595,384
Software Development Aids	COMPILE-ORDER CROSS-REFERENCE MANAGEMENT-TOOLS METRICS PAGER PRETTY-PRINTERS STUBBER SYTLE	87,124 4,457 323,686 304,729 25,442 94,014 12 227,791	359,990 23,786 513,434 1,390,255 95,528 659,742 81,309 1,595,247
		1,078,855	4,719,291
Communications Software	DDN MESSAGE-HANDLING	30,822 242,840	1,959,099 977,501
		273,662	2,936,600
Graphics and Display Tools	FORMGEN GKS MENU	305,823 322,595 367,593	632,772 1,991,575 450,093
		996,011	3,074,440
Other Topics	BENCHMARKS EDITORS EXTERNAL-TOOLS SPELLER STARTER-KIT TOOLS	73,733 113,372 25,043 387,012 18,409 59,876	302,163 144,099 80,520 1,777,350 83,903 323,340
		677,445	2,711,375
<b>TOTAL</b>		<b>5,014,011</b>	<b>15,037,090</b>

**Table 1**

# Data Abstraction with Modula-2

by Bill Walker and Stephen Alexander

***The ability to separate the definition of a data structure from the details of implementation is important.***

## Introduction

This short article is intended to illustrate the concept of data abstraction via the use of the programming language Modula-2. We accomplish this by presenting a sample program that may prove useful in its own right.

We must understand the notion of data abstraction before we can appreciate the utility of the methods presented here. Let us define a priority queue as a collection of data, along with two operations, *add* and *fetch*, which maintain that data on a smallest-item-first (or largest-item-first) basis.

Suppose we have a magic jar that we are free to add or remove data items to or from any time we please. The data we add to the jar is randomly ordered. When we reach into the jar and grab a data item, we are guaranteed that the item will be the smallest data item in the jar. It may not be the smallest that we have seen because we have no idea what data has already been removed from or added to the jar. Our magic jar is acting as a priority queue.

Of major importance is the fact that we have no knowledge of the mechanism of the jar. Internally we do not know if the jar operates by maintaining a computer sorting procedure that is in constant operation or by keeping an especially industrious collection of elves. Nor do we care.

We are interested only in the action of the jar itself, not in its internal mechanism. We may regard the jar

as an abstraction of a priority queue.

We should be aware that the jar has many uses. We can use it as a sorter, for example, by dumping a whole collection of numbers into it and then fetching them one at a time.

## Languages to Support Data Abstraction

Almost any computer language in use today will allow us to write a computer program that will implement a priority queue either by using a static array or by creating a dynamically allocated linked list. The problem with most of these languages is that the method of handling the priority queue is intimately wedded to the abstraction of the queue. There is no convenient method of making a priority queue available to a user without also making the unnecessary details of the implementation available. A few languages, or more commonly language implementations, represent exceptions.

The ability to separate the definition of a data structure from the details of implementation is an important one. This ability is present in several modern languages and in

particular is present in the language Modula-2, which has recently received considerable attention.

## An Example

Modular design charts as presented in *Software Engineering with Modula-2 and Ada* by Sincovec and Wiener<sup>1</sup> represent a natural expression of abstract solutions to software problems. It follows that modular design charts are particularly well suited for use in projects employing programming languages that allow high levels of abstraction. Figure 1, page 63, illustrates a modular design chart of a small software system.

The system represented will be used to generate random numbers and sort them using a priority queue. By examining the chart, we see that the system will consist of a module to generate the pseudorandom numbers; a module to provide the means for handling our abstract priority queue; and a program, Sorter, to tie together the functions supplied by our small "library."

Note that the software bus details the pathways of communication required by the separate modules of our system. The software bus is analogous to a hardware communications bus found on modern computers.

In the design of our chart, we have been careful to minimize the communication required between the modules. By minimizing communication, we have contributed greatly to the dependability and maintainability of our system.

We present a Modula-2 program that follows the scheme of our modular design chart. Our program will generate 100 pseudorandom num-

bers and place them each into a priority queue. After the queue is built, we will remove the numbers from the structure and print them one at a time. The definition of a priority queue assures us that the numbers will be in ascending (or perhaps descending) order. The step from the modular design chart to actual Modula-2 code is often a surprisingly small one.

In this example program, we have "imported" the functions that are designed to deal with the priority queue. We consider that this program is a client of MODULE Queues, which supplies required queue-handling functions. When we write the program, we are entirely ignorant of the method used to implement the priority queue. We can make the same comments about the random number generator *randu()* and its parent module RandomNumbers. We have no idea at all how it works, just that it does. Notice that the program is a direct reflection of the abstract design.

The two modules RandomNumbers and PriorityQueue provide procedures that support the MODULE Sorter (Listing One, page 94). The following section provides a rather complete description of the information a programmer needs to make use of these two modules.

### The Definition Modules

Let us first turn our attention to DEFINITION MODULE RandomNumbers (Listing Two, page 94). In this module, we find that a function procedure named *randu* is exported (made visible) so that other modules can use it. We see that the function procedure *randu* does not require any parameters and that the result type is INTEGER. We have been able to glean all this information by examining the definition module alone. It was not necessary to have any knowledge at all of how the random number generator does its job. The module RandomNumbers is able to provide services to its clients while masking (protecting!) its own internal functions from those clients.

In a like manner, DEFINITION MODULE Queues provides us with all the tools we need to utilize priority queues without providing any access to the internal workings of those

queues. This module exports the type PriorityQueue so that user programs may declare compatible objects of this type. It also provides a means of initializing such queues (PROCEDURE InitPriorityQueue) and for adding items to a queue or fetching an item from a queue. There is also a function procedure that can determine if a given queue is empty. (In a production environment, MODULE Queues would probably contain additional queue-handling procedures. For purposes of illustration, this subset should be sufficient.)

These procedures provide us with a system for manipulating priority queues. MODULE Sorter is an example of how we can use the objects from RandomNumbers and Queues to make a useful program.

Please note that in most recent versions of Modula-2, the EXPORT lists are not required. In these newer versions, the declarations themselves serve as the EXPORT list for the definition modules<sup>2</sup>. In MacModula-2 (from Modula Corporation), the EXPORT lists are required.

### Compiling the Programs

Modula-2 allows for the compilation of both the definition modules and the client program Sorter without concern for the details of implementation. This is important because it al-

lows us to detect any errors in the design of the interface of our system. For instance, if Sorter incorrectly provides us with the wrong number of parameters to one of the imported procedures, the compiler will supply us with ample warning. This feature is in contrast to such languages as C, which would not flag such an error.

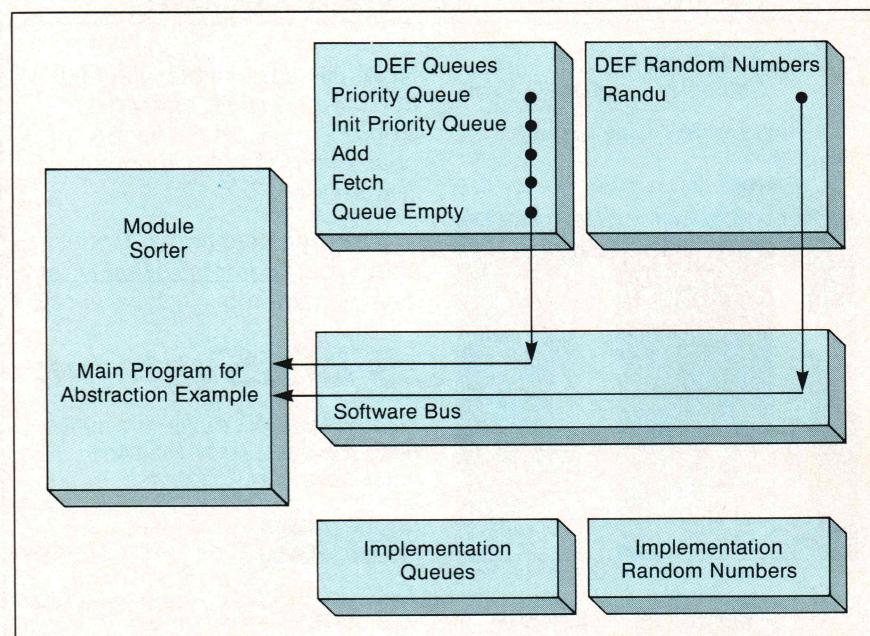
The compiler is sufficiently sophisticated to allow explicit type checking across the boundaries of separately compiled files. It is commonplace to compile each of the definition modules individually and to provide them as libraries for the use of the main program.

At this point, we compiled each of the above-listed programs under MacModula-2 on a 512K Macintosh. It is necessary that both definition modules be compiled before MODULE Sorter because Sorter is a client of these modules. The order of compilation of the definition modules is arbitrary because neither is dependent upon the other.

### Implementation Issues

In this section we provide implementation modules for each of the above-defined modules RandomNumbers and Queues.

A library module consists of an interface (definition module) and an implementation (implementation



**Figure 1**

module). After compiling the implementation modules, we have complete library modules that are available to any client. In production environments it is important that system maintenance be convenient. Modula-2's library concept allows for this needed convenience.

Now we will discuss the implementation components of our small library and illustrate the exceptional ease of maintenance afforded by Modula-2.

Note that the procedures and modules listed in Listings Three and Four (page 95) that are not discussed are part of a standard library of interfaces provided with the MacModula-2 system. It is necessary to import the utilities Allocate and Deallocate from storage in order to use the standard procedures New and Dispose. A discussion of these standard utilities may be found in Niklaus Wirth's *Programming in Modula-2*<sup>2</sup>. We feel that

the mnemonic value of these declarations is great enough that further discussion of them is not required.

We have chosen to implement Queues via the mechanism of a dynamically allocated linked-list structure. Upon examining the implementations, we see first that the code is easily read and much akin to Pascal. Upon closer examination we discern that the Add procedure is a well-known recursively defined algorithm. At this point we recognize that we might later wish to change this procedure to a nonrecursive algorithm. If we do so, it will not be necessary to recompile either the definition modules or the MODULE Sorter. The only recompilation necessary will be that of IMPLEMENTATION MODULE Queues.

The random number generator in IMPLEMENTATION MODULE Random-Numbers is an example of the classic linear congruential multiplicative pseudorandom number generator of the type discussed in Knuth's text<sup>3</sup>, among others. However, we have

chosen the coefficients for the recurrence relation poorly. If we recognize this, say after using MODULE Sorter for six months or so in a production environment, we can simply replace the coefficients with more appropriate ones and recompile the IMPLEMENTATION MODULE Random-Numbers. Because it is not necessary to recompile Sorter, a user of the system might be totally unaware that we have made any changes.

If we find that the uses of our system are not well suited to recursion, we may change PROCEDURE Add of IMPLEMENTATION MODULE Queues to a nonrecursive routine. Once again, it will be necessary to recompile only IMPLEMENTATION MODULE Queues to effect the change.

Before the changes were made, our system was capable of sorting fewer than 1,000 pseudorandom numbers; however, after we installed the nonrecursive version of Add, we used the system to sort more than 4,000 random numbers. (This feat was accomplished overnight. Insertion sorting is as slow sorting method!) In addition, the range of the pseudorandom numbers generated by *randu* was increased from 0 .. 6 to 0 .. 4095.

### Conclusion

In addition to the features of the language shown here, Modula-2 also offers procedure types, generics, open array parameters, low-level facilities, concurrent processes, and many other resources too numerous to mention in this article. Those readers interested in further study of Modula-2 are invited to examine the following list of sources.

### Notes

1. Sincovec, R., and Wiener, R. *Software Engineering with Modula-2 and Ada*. New York: John Wiley and Sons, 1985.
2. Wirth, N. *Programming in Modula-2, Third Edition*. New York: Springer-Verlag, 1985.
3. Knuth, D. E. *The Art of Computer Programming*, vol. 2, *Seminumerical Algorithms*. Reading, Mass.: Addison-Wesley, 1981.

**(Listings begin on page 94)**

#### Reader Ballot

Vote for your favorite feature/article.  
Circle Reader Service No. 6.

Circle no. 78 on reader service card.

**THE BEST Z80  
ASSEMBLER ON  
THE MARKET JUST  
GOT BETTER!**

**Z80ASM**  
NOW \$49.95  
ONLY

**DON'T ASK HOW OURS CAN BE SO FAST...  
ASK WHY THEIRS ARE SO SLOW!**

“...a breath of fresh air...”

Computer Language, Feb. 85

“...in two words, I'd say speed & flexibility”

Edward Joyce, User's Guide #15

Now fully compatible with M80 in Z80 mode with many extensions. Time & date in listing, 16 char. externals, plus many other features.

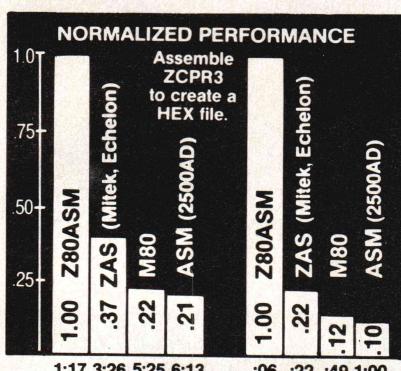
To order, or to find out more about our complete family of development tools, call or write:

**SLR Systems**

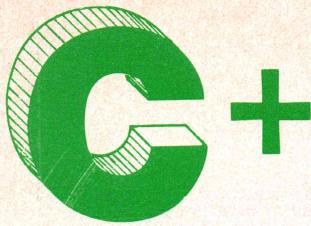
1622 N. Main St., Butler, PA 16001  
(800) 833-3061, (412) 282-0864  
Telex 559215 SLR SYS



C.O.D., Check or  
Money Order Accepted



SHIPPING: USA/CANADA + \$3 • OTHER AREAS + \$10  
Z80 CP/M compatibility required.



# **The Best C Book**

## **A Powerful C Compiler**

---

### **One Great C Value \$39.95**

A good C book just isn't complete without a good C compiler to go with it. That's why we give you both. You get a comprehensive 450 page book and a full feature standard K&R C compiler with the Unix V7 Extensions. The Book is loaded with examples that teach you how to program in C. And our fast one pass C compiler comes with an equally fast

linker so you don't waste a lot of time watching your disk drives spin. You also get a Unix compatible function library that contains more than 150 functions (C source code included). And if all that isn't enough, we offer you a 30 day money back guarantee. So what are you waiting for? The exciting world of C is just one free phone call away.

## Language Features

- Data Types: char, short, int, unsigned, long, float, double
- Data Classes: auto, extern, static, register
- Typedef, Struct, Union, Bit Fields, Enumerations
- Structure Assignment, Passing/Returning Structures

abs	confuf	feof
asm	conc	ferro
asmx	cos	flush
atan	cpystr	fgets
atof	creat	fileno
atoi	cursblk	filetrat
atol	curslin	find
bdbos	curscol	floor
bdbosx	cursorw	fopen
bios	cursoff	fprintf
biosx	curson	iputs
calloc	delete	freadd
ceil	drand	free
cfree	exec	freopen
chain	execel	fsfscanf
character	execv	fseek
chdir	exit	ftell
chmod	exitmsg	fwrite
clearerr	exp	getc
close	fabs	getch
clrscrn	fclose	putchar
cmon	filenon	getchar

## Functions

getcseg	isascii	movmem	replace	strcat
getdseg	iscntrl	open	repmem	strncpy
getd	isdigit	outp	rewind	strcp
putd	islower	peek	right\$	strlen
getdate	isprint	perror	rindex	strncat
gettime	ispunct	poke	rmdir	strncmp
geti	isspace	poscurs	scnf	strncpy
puti	isupper	pow	setbuf	strsave
getkey	itoa	printf	setbufsiz	system
getmode	keypress	putc	setcolor	tolower
setmode	left\$	putchar	setdate	toupper
gets	len	puts	settime	ungetc
getw	log	putw	setjmp	ungetch
heapsz	log10	rand	setmem	unlink
heatrap	longjmp	read	sin	write
hypot	lseek	readattr	sound	writechs
index	malloc	reach	sprintf	xmembeg
inp	alloc	writech	sqrt	xmemend
insert	mathtrap	readdot	strand	xmemget
iofilter	mid\$	writtenot	sscanf	xmemput
isalnum	mkdir	realloc	stacksiz	xmovmem
isalnum	modf	rename	str\$	_exit
isalpcha				

## **MIX Editor**

**\$29.95**

When you're programming in a high level language you need a high powered editor. That's why we created a programmable full/split screen text processor. It lets you split the screen horizontally or vertically and edit two files at once. You can move text back and forth between two windows. You can also create your own macro commands from an assortment of over

100 predefined commands. The editor comes configured so that it works just like Wordstar but you can change it if you prefer a different keyboard layout. The editor is a great companion to our C compiler. Because they work so well together we want you to have both. To make sure you do, we're offering the editor for just \$15 when purchased with the C compiler.

## ASM Utility \$10

The ASM utility disk allows you to link object files created by Microsoft's MASM or M80 assemblers. Lots of useful assembly language functions are included as examples.

**ORDERS ONLY**  
**1-800-523-9520**  
**IN TEXAS**  
**1-800-622-4070**

**Canadian Distributor**  
**Saraguay Software: 416-923-1500**

**NOT COPY PROTECTED**

Editor	\$ _____	(29.95)
C	\$ _____	(39.95)
C & Editor	\$ _____	(54.95)
ASM Utility	\$ _____	(10.00)
TX Residents	\$ _____	(6.125% sales tax)
Shipping	\$ _____	(see below)
<b>Total</b>	\$ _____	
<input type="checkbox"/> Check	<input type="checkbox"/> Money Order	
<input type="checkbox"/> MC/Visa*	_____	Exp _____
<b>Shipping Charges:</b> (No charge for ASM Utility)		
USA: \$5/Order		
Canada: \$10/Order		
Overseas: \$10/Editor • \$20/C • \$30/C & Editor		

- PCDOS/MSDOS (2.0 or later)
  - IBM PC Single Side
  - IBM PC Double Side
  - Tandy 2000
  - 8 Inch
  - Other \_\_\_\_\_
- CPM 80 (2.2 or later)
  - 8 Inch
  - Kaypro II
  - Kaypro 4
  - Apple (Z80)
  - Osborne I SD
  - Osborne I DD
  - Morrow MD II
  - Other \_\_\_\_\_

Name \_\_\_\_\_

Street \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_

Zip \_\_\_\_\_

Country \_\_\_\_\_

Phone \_\_\_\_\_

**MDX**  
software  
(214) 783-6001

2116 E. Arapaho  
Suite 363  
Richardson, TX 75081

Ask about our volume discounts.

# Instant-C™ The Fastest Interpreter for C

Runs your programs 50 to 500 times faster than any other C language interpreter.

Any C interpreter can save you compile and link time when developing your programs. But only **Instant-C** saves your time by running your program at compiled-code speed.

**Fastest Development.** A program that runs in one second when compiled with an optimizing compiler runs in two or three seconds with **Instant-C**. Other interpreters will run the same program in two minutes. Or even ten minutes. Don't trade slow compiling and linking for slow testing and debugging. *Only Instant-C will let you edit, test, and debug at the fastest possible speeds.*

**Fastest Testing.** **Instant-C** immediately executes any C expression, statement, or function call, and display the results. Learn C, or test your programs faster than ever before.

**Fastest Debugging.** **Instant-C** gives you the best source-level debugger for C. Single-step by source statement, or set any number of conditional breakpoints throughout your program. Errors always show the source statements involved. Once you find the problem, test the correction in seconds.

**Fastest Programming.** **Instant-C** can directly generate executable files, supports full K & R standard C, comes with complete library source, and works under PC-DOS, MS-DOS, or CP/M-86.

**Instant-C** gives you working, well-tested programs faster than any other programming tool. Satisfaction guaranteed, or your money back in first 31 days. **Instant-C** is \$495.

**Rational**  
Systems, Inc.

P.O. Box 480  
Natick, MA 01760  
(617) 653-6194

## C CHEST

### LISTING TWO (Text begins on page 16)

```
1 #include <stdio.h>
2 #include <ctype.h>
3
4 /*      HIST.C  Support for Unix-like history.  In addition, if ! is
5 * replaced with a ^ you'll be in edit mode before
6 * re-executing the command.  Legal syntaxes are:
7 *
8 *      Copyright (C) 1985 Allen I. Holub.  All rights reserved.
9 */
10
11 *      non-edit      edit      function
12 *      mode:        mode:    performed:
13 *
14 *          !!          ^^      repeat previous command
15 *          !<num>      ^<num>  repeat cmd with hist number <num>
16 *          !<pat>      ^<pat>  repeat cmd that matches pat.
17 *          !> file      ^< file  Save history list to file
18 *          !< file      ^> file  Restore history list from file
19
20 *      Externally accessible routines are:
21 *
22 *      void      print_hist(fp)  Print the history list along with
23 *      FILE     *fp;           associated nums (for the !<num> command).
24 *
25 *      int       get_hnum()    Returns the history number that will be
26 *                           associated with the next call to history.
27 *
28 *      void      history( buf, maxbuf)
29 *      char     *buf;          Add the contents of buf to the history
30 *                           list.  If buf contains a history request
31 *                           (! etc.) expand it too.  In this case,
32 *                           overwrite buf with the entry from the
33 *                           history list.  No more than maxbuf
34 *                           characters will be copied.
35 */
36 */
37 */
38
39 extern char *strsave ( char* );
40 extern int strmatch ( char*, char* );
41 extern char *efgets ( char*, int, FILE* );
42
43 */
44 */
45 #define MAXHIST 32           /* Size of history list */
46 #define START Hist_list;
47 #define END   &Hist_list[MAXHIST-1]
48 #define HIST_CH '!'
49 #define EDIT_CH '^'
50 #define READ_CH '<'
51 #define WRITE_CH '>'
52
53 #define DEF_HNAME "/histlist" /* Default place to save history list with
54 *                           * !> or !< commands. (ie. no file given).
55 */
56 */
57 /* History list, maintained as a ring buffer: */
58
59 static char *Hist_list[MAXHIST] = {
60 {
61     NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
62     NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
63     NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
64     NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL
65 };
66
67 static char **Hist_ptr = START; /* Pointer into history list */
68 static int Hist_num = 1; /* Current history number */
69
70 */
71
72 static void add_hist( cmd )
73 char *cmd;
74 {
75     /* If cmd is non-null, add it to the history list.
76     * Commands are copied into a mallocoed buffer.
77     */
78
79     if( *cmd )
80     {
81         if( *Hist_ptr )
82             free( *Hist_ptr );
83
84         if( ! (*Hist_ptr = strsave(cmd)) )
85             printf("Out of memory !!\n");
86         else
87         {
88             Hist_num++;
89             if( !Hist_ptr > END )
90                 Hist_ptr = START;
91         }
92     }
93 }
94
95 */
96
97 static int get_hist(buf, maxbuf, desired)
98 register int desired;
99 char *buf;
100 {
101     /* Copy the string associated with the indicated history
102     * number into buf if possible.  Return 1 on success & 0 if
103     * the number isn't in the list (is too old).  Negative
104     * numbers are treated as an offset from the current
105     * history number.  If "desired" is 0 then we are
106     * searching for a pattern rather than getting a particular
107     * number.  In this case the pattern starts at buf[1];
108 }
```

```

108     */
109
110     register char **p;
111
112     if( !desired )
113     {
114         /*      A pattern was requested. Return 0 if you can't
115         *      find it. Otherwise, when the loop is done *p will
116         *      point at the correct string. Search the list
117         *      backwards starting at the most recently entered
118         *      number.
119         */
120
121         for( p = Hist_ptr, ++buf ; 1 ; )
122         {
123             p = (p == START) ? END : p - 1 ;
124             if( !strmatch(buf, *p) )
125                 break;
126
127             if( p == Hist_ptr || !*p )
128                 return 0;
129         }
130
131         --buf;
132     }
133
134     else
135     {
136         /*      An actual history number has been requested:
137         *      1)      Convert a negative offset into an actual
138         *              history number if necessary.
139         *      2)      Then convert the history number into an
140         *              offset into the history list. Note that
141         *              this will be a negative offset even though
142         *              the result of this operation is positive.
143         *      3)      If the offset is out of range, return a 0.
144         *      4)      Else convert the offset into a pointer into
145         *              the history list. If the pointer is off the
146         *              list wrap around to the other end.
147         */
148
149         if( desired < 0 )                                /* 1 */
150             desired += Hist_num;
151
152         desired = Hist_num - desired;                   /* 2 */
153
154         if( desired < 1 || desired > MAXHIST )          /* 3 */
155             return 0;
156
157         if( (p = Hist_ptr - desired) < START )          /* 4 */
158             p += MAXHIST;
159     }
160
161     strncpy( buf, *p, maxbuf );
162
163     return 1;
164 }
165
166 */
167
168 static int    eget_hist( buf, maxbuf, desired )
169 char    *buf;
170 {
171     /*      Works just like get_hist except lets you edit the
172     *      command. Note that the string returned by this routine
173     *      will be overwritten by the next call. Copy it somewhere
174     *      if you need to save it.
175     */
176
177     register int    rval;
178
179     if( rval = get_hist(buf, maxbuf, desired) )
180         rval = - (int) efgets(buf, maxbuf, stdin) ;
181
182     return rval ;
183 }
184
185 */
186
187 int    get_hnum()
188 {
189     return Hist_num ;
190 }
191
192 */
193
194 void    print_hist( stream )
195 FILE    *stream;
196
197 {
198     register char **hp = Hist_ptr ;
199     register int    i = Hist_num - MAXHIST ;
200
201     do {
202         if( i >= 1 )
203             fprintf(stream, "%3d: %s\n", i, *hp );
204
205         if( ++hp > END )
206             hp = START;
207
208     } while( ++i < Hist_num ) ;
209 }
210
211
212 */
213
214 char    *hist_name( fname )
215 register char    *fname;
216
217     /*      Strip leading white space from fname and then
218     *      return a pointer to either the first non-white
219     *      character or to DEF_HNAME if there is no
220     *      non-white character

```

(Continued on next page)

# SUPER DIR

★★ FEATURES ★★  
THAT DOS FORGOT!

/AFTER = DATE [TIME]  
/BEFORE = DATE [TIME]  
/EXCLUDE = (FILE SPEC)  
/INCLUDE = (FILE SPEC)  
/[NO] SIZE  
/[NO] DATE  
/[NO] PATH  
/[NO] STATS  
/BRIEF

FOR MS/PC-DOS  
VER 2.0 - LATER

**\$24.95**

Plus Shipping

California Residents  
Please Add 6% Sales Tax

VISA / MASTER CARD



MONEY ORDER / C.O.D.  
Checks Allow Two Weeks

FOR INFORMATION OR ORDERS  
CALL:

**714/496-8599**

9:00 A.M. - 5:00 P.M. Pacific Time

OR SEND ORDER TO:

**DRAFTRONICS**  
34184 COAST HIGHWAY  
SUITE #198  
DANA POINT,  
CALIFORNIA 92629

Circle no. 235 on reader service card.

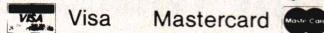
# PRIME FEATURES

- Execute DOS level commands in HS/FORTH, or execute DOS and BIOS functions directly.
- Execute other programs under HS/FORTH supervision.  
(editors debuggers file managers etc)
- Use our editor or your own.
- Save environment any time as .COM or .EXE file.
- Eliminate headers, reclaim space without recompiling.
- Trace and decompile.
- Deferred definition, execution vectors, case, interrupt handlers.

# HS / FORTH

- Full 8087 high level support. Full range transcendents (tan sin cos arctan logs exponentials)
- Data type conversion and I/O parse/format to 18 digits plus exponent.
- Complete Assembler for 8088, 80186, and 8087.
- String functions -  
(LEFT RIGHT MID LOC COMP XCHG JOIN)
- Graphics & Music
- Includes Forth-79 and Forth-83
- File and/or Screen interfaces
- Segment Management
- Full megabyte - programs or data
- Fully Optimized & Tested for:  
IBM-PC XT AT and JR  
COMPAQ and TANDY 1000 & 2000  
(Runs on all true MSDOS compatibles!)
- Compare  
BYTE Sieve Benchmark jan 83  
HS/FORTH 47 sec BASIC 2000 sec  
with AUTO-OPT 9 sec Assembler 5 sec  
other Forths (mostly 64k) 55-140 sec  
FASTEST FORTH SYSTEM  
AVAILABLE.  
TWICE AS FAST AS OTHER  
FULL MEGABYTE FORTHS!  
(TEN TIMES FASTER WHEN USING AUTO-OPT)

HS/FORTH, complete system only: \$270.



# HARVARD SOFTWORKS

P.O. BOX 69  
SPRINGBORO, OH 45066  
(513) 748-0390

Circle no. 132 on reader service card.

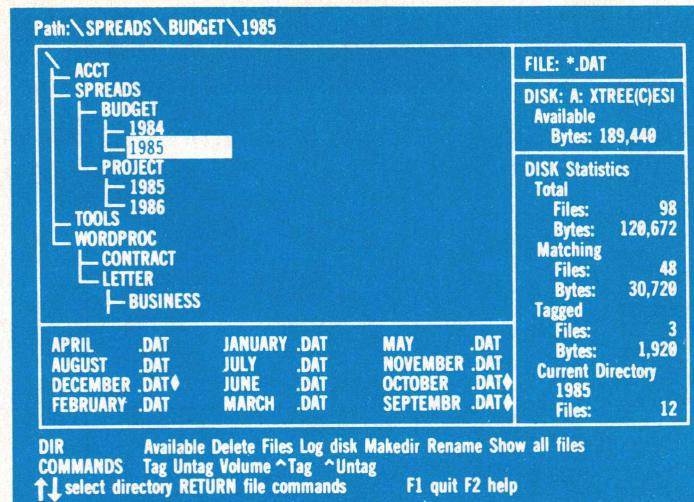
# C CHEST

## LISTING TWO (Listing continued, text begins on page 16)

```

221      */
222      while( isspace(*fname) )
223          fname++;
224
225      return ( *fname ) ? fname : DEF_HNAME ;
226  }
227  /*-----
228
229  231 save hist( fname )
232  register char  *fname;
233  {
234      /*      Save the current history list in either the indicated
235      *      file or in DEF_HNAME if no file is given.
236      */
237
238      register FILE  *fp;
239
240      fname = hist_name( fname );
241
242      if( !(fp = fopen( fname, "w" )) )
243          perror( fname );
244      else
245      {
246          print_hist(fp);
247          fclose(fp);
248      }
249  }
250  /*-----
251
252  253 restore hist( fname )
254  register char  *fname;
255  {
256      /*      Add the contents of fname (or DEF_HNAME if fname is
257      *      empty) to the history list. Don't execute the
258      *      commands.
259      */
260
261      register FILE  *fp ;
262      register char  *p ;
263      char          *buf ;
264
265      fname = hist_name( fname );
266
267      if( !(fp = fopen( fname, "r" )) )
268          perror( fname );
269
270      else if( !(buf = malloc(1024) ) )
271          fprintf(stderr,"Not enough memory\n");
272      else
273      {
274          while( efgets(buf, 1024, fp) )
275          {
276              for(p = buf; isdigit(*p) || isspace(*p); p++ )
277
278                  if( *p == ':' )           /* Skip a :<blank> if      */
279                      p++;                /* one is present      */
280
281                  if( *p == ',' )
282                      p++;
283
284          add_hist( p );
285      #ifdef DEBUG
286          printf("adding %3d: %s\n", Hist_num - 1, p );
287      #endif
288      }
289
290      free( buf );
291  }
292  }
293  /*-----
294
295  296 void      history( buf, maxbuf )
296  register char  *buf;
298  {
299      register int   i = 1 ;
300      register char  *p = buf;
301
302      /*
303      *      If buf contains a history request (!! ^<num> or ^<num>)
304      *      replace its contents as indicated. In any event add the
305      *      contents of buf (after the replacement if one was done)
306      *      to the history list. Blank lines are ignored. If a history
307      *      expansion occurs, the expanded line is printed.
308      *      Null strings won't be added to the history list.
309      */
310
311      if( !*p )
312          return;
313
314      if( *p == HIST_CH || *p == EDIT_CH )
315      {
316          /* First increment p to point at the second character
317          * in the buffer. If the second character in the buffer is
318          * a ! or ^ then i = -1, otherwise it gets the history #.
319          */
320
321          if( ++p == READ_CH )
322          {
323              restore_hist( p + 1 );
324              *buf = '\0';
325          }
326          else if( *p == WRITE_CH )
327          {
328              save_hist( p + 1 );
329              *buf = '\0';
330          }
331
332      }
333
334  }
335
336  /*
337  *      If the buffer is empty, then the history list is
338  *      empty. If the buffer is not empty, then the history list
339  *      is not empty.
340  */
341
342  /*
343  *      If the buffer is empty, then the history list is
344  *      empty. If the buffer is not empty, then the history list
345  *      is not empty.
346  */
347
348  /*
349  *      If the buffer is empty, then the history list is
350  *      empty. If the buffer is not empty, then the history list
351  *      is not empty.
352  */
353
354  /*
355  *      If the buffer is empty, then the history list is
356  *      empty. If the buffer is not empty, then the history list
357  *      is not empty.
358  */
359
360  /*
361  *      If the buffer is empty, then the history list is
362  *      empty. If the buffer is not empty, then the history list
363  *      is not empty.
364  */
365
366  /*
367  *      If the buffer is empty, then the history list is
368  *      empty. If the buffer is not empty, then the history list
369  *      is not empty.
370  */
371
372  /*
373  *      If the buffer is empty, then the history list is
374  *      empty. If the buffer is not empty, then the history list
375  *      is not empty.
376  */
377
378  /*
379  *      If the buffer is empty, then the history list is
380  *      empty. If the buffer is not empty, then the history list
381  *      is not empty.
382  */
383
384  /*
385  *      If the buffer is empty, then the history list is
386  *      empty. If the buffer is not empty, then the history list
387  *      is not empty.
388  */
389
390  /*
391  *      If the buffer is empty, then the history list is
392  *      empty. If the buffer is not empty, then the history list
393  *      is not empty.
394  */
395
396  /*
397  *      If the buffer is empty, then the history list is
398  *      empty. If the buffer is not empty, then the history list
399  *      is not empty.
400  */
401
402  /*
403  *      If the buffer is empty, then the history list is
404  *      empty. If the buffer is not empty, then the history list
405  *      is not empty.
406  */
407
408  /*
409  *      If the buffer is empty, then the history list is
410  *      empty. If the buffer is not empty, then the history list
411  *      is not empty.
412  */
413
414  /*
415  *      If the buffer is empty, then the history list is
416  *      empty. If the buffer is not empty, then the history list
417  *      is not empty.
418  */
419
420  /*
421  *      If the buffer is empty, then the history list is
422  *      empty. If the buffer is not empty, then the history list
423  *      is not empty.
424  */
425
426  /*
427  *      If the buffer is empty, then the history list is
428  *      empty. If the buffer is not empty, then the history list
429  *      is not empty.
430  */
431
432  /*
433  *      If the buffer is empty, then the history list is
434  *      empty. If the buffer is not empty, then the history list
435  *      is not empty.
436  */
437
438  /*
439  *      If the buffer is empty, then the history list is
440  *      empty. If the buffer is not empty, then the history list
441  *      is not empty.
442  */
443
444  /*
445  *      If the buffer is empty, then the history list is
446  *      empty. If the buffer is not empty, then the history list
447  *      is not empty.
448  */
449
450  /*
451  *      If the buffer is empty, then the history list is
452  *      empty. If the buffer is not empty, then the history list
453  *      is not empty.
454  */
455
456  /*
457  *      If the buffer is empty, then the history list is
458  *      empty. If the buffer is not empty, then the history list
459  *      is not empty.
460  */
461
462  /*
463  *      If the buffer is empty, then the history list is
464  *      empty. If the buffer is not empty, then the history list
465  *      is not empty.
466  */
467
468  /*
469  *      If the buffer is empty, then the history list is
470  *      empty. If the buffer is not empty, then the history list
471  *      is not empty.
472  */
473
474  /*
475  *      If the buffer is empty, then the history list is
476  *      empty. If the buffer is not empty, then the history list
477  *      is not empty.
478  */
479
480  /*
481  *      If the buffer is empty, then the history list is
482  *      empty. If the buffer is not empty, then the history list
483  *      is not empty.
484  */
485
486  /*
487  *      If the buffer is empty, then the history list is
488  *      empty. If the buffer is not empty, then the history list
489  *      is not empty.
490  */
491
492  /*
493  *      If the buffer is empty, then the history list is
494  *      empty. If the buffer is not empty, then the history list
495  *      is not empty.
496  */
497
498  /*
499  *      If the buffer is empty, then the history list is
500  *      empty. If the buffer is not empty, then the history list
501  *      is not empty.
502  */
503
504  /*
505  *      If the buffer is empty, then the history list is
506  *      empty. If the buffer is not empty, then the history list
507  *      is not empty.
508  */
509
510  /*
511  *      If the buffer is empty, then the history list is
512  *      empty. If the buffer is not empty, then the history list
513  *      is not empty.
514  */
515
516  /*
517  *      If the buffer is empty, then the history list is
518  *      empty. If the buffer is not empty, then the history list
519  *      is not empty.
520  */
521
522  /*
523  *      If the buffer is empty, then the history list is
524  *      empty. If the buffer is not empty, then the history list
525  *      is not empty.
526  */
527
528  /*
529  *      If the buffer is empty, then the history list is
530  *      empty. If the buffer is not empty, then the history list
531  *      is not empty.
532  */
533
534  /*
535  *      If the buffer is empty, then the history list is
536  *      empty. If the buffer is not empty, then the history list
537  *      is not empty.
538  */
539
540  /*
541  *      If the buffer is empty, then the history list is
542  *      empty. If the buffer is not empty, then the history list
543  *      is not empty.
544  */
545
546  /*
547  *      If the buffer is empty, then the history list is
548  *      empty. If the buffer is not empty, then the history list
549  *      is not empty.
550  */
551
552  /*
553  *      If the buffer is empty, then the history list is
554  *      empty. If the buffer is not empty, then the history list
555  *      is not empty.
556  */
557
558  /*
559  *      If the buffer is empty, then the history list is
560  *      empty. If the buffer is not empty, then the history list
561  *      is not empty.
562  */
563
564  /*
565  *      If the buffer is empty, then the history list is
566  *      empty. If the buffer is not empty, then the history list
567  *      is not empty.
568  */
569
570  /*
571  *      If the buffer is empty, then the history list is
572  *      empty. If the buffer is not empty, then the history list
573  *      is not empty.
574  */
575
576  /*
577  *      If the buffer is empty, then the history list is
578  *      empty. If the buffer is not empty, then the history list
579  *      is not empty.
580  */
581
582  /*
583  *      If the buffer is empty, then the history list is
584  *      empty. If the buffer is not empty, then the history list
585  *      is not empty.
586  */
587
588  /*
589  *      If the buffer is empty, then the history list is
590  *      empty. If the buffer is not empty, then the history list
591  *      is not empty.
592  */
593
594  /*
595  *      If the buffer is empty, then the history list is
596  *      empty. If the buffer is not empty, then the history list
597  *      is not empty.
598  */
599
599  /*
600  *      If the buffer is empty, then the history list is
601  *      empty. If the buffer is not empty, then the history list
602  *      is not empty.
603  */
603
604  /*
605  *      If the buffer is empty, then the history list is
606  *      empty. If the buffer is not empty, then the history list
607  *      is not empty.
608  */
608
609  /*
610  *      If the buffer is empty, then the history list is
611  *      empty. If the buffer is not empty, then the history list
612  *      is not empty.
613  */
613
614  /*
615  *      If the buffer is empty, then the history list is
616  *      empty. If the buffer is not empty, then the history list
617  *      is not empty.
618  */
618
619  /*
620  *      If the buffer is empty, then the history list is
621  *      empty. If the buffer is not empty, then the history list
622  *      is not empty.
623  */
623
624  /*
625  *      If the buffer is empty, then the history list is
626  *      empty. If the buffer is not empty, then the history list
627  *      is not empty.
628  */
628
629  /*
630  *      If the buffer is empty, then the history list is
631  *      empty. If the buffer is not empty, then the history list
632  *      is not empty.
633  */
633
634  /*
635  *      If the buffer is empty, then the history list is
636  *      empty. If the buffer is not empty, then the history list
637  *      is not empty.
638  */
638
639  /*
640  *      If the buffer is empty, then the history list is
641  *      empty. If the buffer is not empty, then the history list
642  *      is not empty.
643  */
643
644  /*
645  *      If the buffer is empty, then the history list is
646  *      empty. If the buffer is not empty, then the history list
647  *      is not empty.
648  */
648
649  /*
649  *      If the buffer is empty, then the history list is
650  *      empty. If the buffer is not empty, then the history list
651  *      is not empty.
652  */
652
653  /*
654  *      If the buffer is empty, then the history list is
655  *      empty. If the buffer is not empty, then the history list
656  *      is not empty.
657  */
657
658  /*
659  *      If the buffer is empty, then the history list is
660  *      empty. If the buffer is not empty, then the history list
661  *      is not empty.
662  */
662
663  /*
664  *      If the buffer is empty, then the history list is
665  *      empty. If the buffer is not empty, then the history list
666  *      is not empty.
667  */
667
668  /*
669  *      If the buffer is empty, then the history list is
670  *      empty. If the buffer is not empty, then the history list
671  *      is not empty.
672  */
672
673  /*
674  *      If the buffer is empty, then the history list is
675  *      empty. If the buffer is not empty, then the history list
676  *      is not empty.
677  */
677
678  /*
679  *      If the buffer is empty, then the history list is
680  *      empty. If the buffer is not empty, then the history list
681  *      is not empty.
682  */
682
683  /*
684  *      If the buffer is empty, then the history list is
685  *      empty. If the buffer is not empty, then the history list
686  *      is not empty.
687  */
687
688  /*
689  *      If the buffer is empty, then the history list is
690  *      empty. If the buffer is not empty, then the history list
691  *      is not empty.
692  */
692
693  /*
694  *      If the buffer is empty, then the history list is
695  *      empty. If the buffer is not empty, then the history list
696  *      is not empty.
697  */
697
698  /*
699  *      If the buffer is empty, then the history list is
700  *      empty. If the buffer is not empty, then the history list
701  *      is not empty.
702  */
702
703  /*
704  *      If the buffer is empty, then the history list is
705  *      empty. If the buffer is not empty, then the history list
706  *      is not empty.
707  */
707
708  /*
709  *      If the buffer is empty, then the history list is
710  *      empty. If the buffer is not empty, then the history list
711  *      is not empty.
712  */
712
713  /*
714  *      If the buffer is empty, then the history list is
715  *      empty. If the buffer is not empty, then the history list
716  *      is not empty.
717  */
717
718  /*
719  *      If the buffer is empty, then the history list is
720  *      empty. If the buffer is not empty, then the history list
721  *      is not empty.
722  */
722
723  /*
724  *      If the buffer is empty, then the history list is
725  *      empty. If the buffer is not empty, then the history list
726  *      is not empty.
727  */
727
728  /*
729  *      If the buffer is empty, then the history list is
730  *      empty. If the buffer is not empty, then the history list
731  *      is not empty.
732  */
732
733  /*
734  *      If the buffer is empty, then the history list is
735  *      empty. If the buffer is not empty, then the history list
736  *      is not empty.
737  */
737
738  /*
739  *      If the buffer is empty, then the history list is
740  *      empty. If the buffer is not empty, then the history list
741  *      is not empty.
742  */
742
743  /*
744  *      If the buffer is empty, then the history list is
745  *      empty. If the buffer is not empty, then the history list
746  *      is not empty.
747  */
747
748  /*
749  *      If the buffer is empty, then the history list is
750  *      empty. If the buffer is not empty, then the history list
751  *      is not empty.
752  */
752
753  /*
754  *      If the buffer is empty, then the history list is
755  *      empty. If the buffer is not empty, then the history list
756  *      is not empty.
757  */
757
758  /*
759  *      If the buffer is empty, then the history list is
760  *      empty. If the buffer is not empty, then the history list
761  *      is not empty.
762  */
762
763  /*
764  *      If the buffer is empty, then the history list is
765  *      empty. If the buffer is not empty, then the history list
766  *      is not empty.
767  */
767
768  /*
769  *      If the buffer is empty, then the history list is
770  *      empty. If the buffer is not empty, then the history list
771  *      is not empty.
772  */
772
773  /*
774  *      If the buffer is empty, then the history list is
775  *      empty. If the buffer is not empty, then the history list
776  *      is not empty.
777  */
777
778  /*
779  *      If the buffer is empty, then the history list is
780  *      empty. If the buffer is not empty, then the history list
781  *      is not empty.
782  */
782
783  /*
784  *      If the buffer is empty, then the history list is
785  *      empty. If the buffer is not empty, then the history list
786  *      is not empty.
787  */
787
788  /*
789  *      If the buffer is empty, then the history list is
790  *      empty. If the buffer is not empty, then the history list
791  *      is not empty.
792  */
792
793  /*
794  *      If the buffer is empty, then the history list is
795  *      empty. If the buffer is not empty, then the history list
796  *      is not empty.
797  */
797
798  /*
799  *      If the buffer is empty, then the history list is
800  *      empty. If the buffer is not empty, then the history list
801  *      is not empty.
802  */
802
803  /*
804  *      If the buffer is empty, then the history list is
805  *      empty. If the buffer is not empty, then the history list
806  *      is not empty.
807  */
807
808  /*
809  *      If the buffer is empty, then the history list is
810  *      empty. If the buffer is not empty, then the history list
811  *      is not empty.
812  */
812
813  /*
814  *      If the buffer is empty, then the history list is
815  *      empty. If the buffer is not empty, then the history list
816  *      is not empty.
817  */
817
818  /*
819  *      If the buffer is empty, then the history list is
820  *      empty. If the buffer is not empty, then the history list
821  *      is not empty.
822  */
822
823  /*
824  *      If the buffer is empty, then the history list is
825  *      empty. If the buffer is not empty, then the history list
826  *      is not empty.
827  */
827
828  /*
829  *      If the buffer is empty, then the history list is
830  *      empty. If the buffer is not empty, then the history list
831  *      is not empty.
832  */
832
833  /*
834  *      If the buffer is empty, then the history list is
835  *      empty. If the buffer is not empty, then the history list
836  *      is not empty.
837  */
837
838  /*
839  *      If the buffer is empty, then the history list is
840  *      empty. If the buffer is not empty, then the history list
841  *      is not empty.
842  */
842
843  /*
844  *      If the buffer is empty, then the history list is
845  *      empty. If the buffer is not empty, then the history list
846  *      is not empty.
847  */
847
848  /*
849  *      If the buffer is empty, then the history list is
850  *      empty. If the buffer is not empty, then the history list
851  *      is not empty.
852  */
852
853  /*
854  *      If the buffer is empty, then the history list is
855  *      empty. If the buffer is not empty, then the history list
856  *      is not empty.
857  */
857
858  /*
859  *      If the buffer is empty, then the history list is
860  *      empty. If the buffer is not empty, then the history list
861  *      is not empty.
862  */
862
863  /*
864  *      If the buffer is empty, then the history list is
865  *      empty. If the buffer is not empty, then the history list
866  *      is not empty.
867  */
867
868  /*
869  *      If the buffer is empty, then the history list is
870  *      empty. If the buffer is not empty, then the history list
871  *      is not empty.
872  */
872
873  /*
874  *      If the buffer is empty, then the history list is
875  *      empty. If the buffer is not empty, then the history list
876  *      is not empty.
877  */
877
878  /*
879  *      If the buffer is empty, then the history list is
880  *      empty. If the buffer is not empty, then the history list
881  *      is not empty.
882  */
882
883  /*
884  *      If the buffer is empty, then the history list is
885  *      empty. If the buffer is not empty, then the history list
886  *      is not empty.
887  */
887
888  /*
889  *      If the buffer is empty, then the history list is
890  *      empty. If the buffer is not empty, then the history list
891  *      is not empty.
892  */
892
893  /*
894  *      If the buffer is empty, then the history list is
895  *      empty. If the buffer is not empty, then the history list
896  *      is not empty.
897  */
897
898  /*
899  *      If the buffer is empty, then the history list is
900  *      empty. If the buffer is not empty, then the history list
901  *      is not empty.
902  */
902
903  /*
904  *      If the buffer is empty, then the history list is
905  *      empty. If the buffer is not empty, then the history list
906  *      is not empty.
907  */
907
908  /*
909  *      If the buffer is empty, then the history list is
910  *      empty. If the buffer is not empty, then the history list
911  *      is not empty.
912  */
912
913  /*
914  *      If the buffer is empty, then the history list is
915  *      empty. If the buffer is not empty, then the history list
916  *      is not empty.
917  */
917
918  /*
919  *      If the buffer is empty, then the history list is
920  *      empty. If the buffer is not empty, then the history list
921  *      is not empty.
922  */
922
923  /*
924  *      If the buffer is empty, then the history list is
925  *      empty. If the buffer is not empty, then the history list
926  *      is not empty.
927  */
927
928  /*
929  *      If the buffer is empty, then the history list is
930  *      empty. If the buffer is not empty, then the history list
931  *      is not empty.
932  */
932
933  /*
934  *      If the buffer is empty, then the history list is
935  *      empty. If the buffer is not empty, then the history list
936  *      is not empty.
937  */
937
938  /*
939  *      If the buffer is empty, then the history list is
940  *      empty. If the buffer is not empty, then the history list
941  *      is not empty.
942  */
942
943  /*
944  *      If the buffer is empty, then the history list is
945  *      empty. If the buffer is not empty, then the history list
946  *      is not empty.
947  */
947
948  /*
949  *      If the buffer is empty, then the history list is
950  *      empty. If the buffer is not empty, then the history list
951  *      is not empty.
952  */
952
953  /*
954  *      If the buffer is empty, then the history list is
955  *      empty. If the buffer is not empty, then the history list
956  *      is not empty.
957  */
957
958  /*
959  *      If the buffer is empty, then the history list is
960  *      empty. If the buffer is not empty, then the history list
961  *      is not empty.
962  */
962
963  /*
964  *      If the buffer is empty, then the history list is
965  *      empty. If the buffer is not empty, then the history list
966  *      is not empty.
967  */
967
968  /*
969  *      If the buffer is empty, then the history list is
970  *      empty. If the buffer is not empty, then the history list
971  *      is not empty.
972  */
972
973  /*
974  *      If the buffer is empty, then the history list is
975  *      empty. If the buffer is not empty, then the history list
976  *      is not empty.
977  */
977
978  /*
979  *      If the buffer is empty, then the history list is
980  *      empty. If the buffer is not empty, then the history list
981  *      is not empty.
982  */
982
983  /*
984  *      If the buffer is empty, then the history list is
985  *      empty. If the buffer is not empty, then the history list
986  *      is not empty.
987  */
987
988  /*
989  *      If the buffer is empty, then the history list is
990  *      empty. If the buffer is not empty, then the history list
991  *      is not empty.
992  */
992
993  /*
994  *      If the buffer is empty, then the history list is
995  *      empty. If the buffer is not empty, then the history list
996  *      is not empty.
997  */
997
998  /*
999  *      If the buffer is empty, then the history list is
1000  *      empty. If the buffer is not empty, then the history list
1001  *      is not empty.
1002  */
1002
1003  /*
1004  *      If the buffer is empty, then the history list is
1005  *      empty. If the buffer is not empty, then the history list
1006  *      is not empty.
1007  */
1007
1008  /*
1009  *      If the buffer is empty, then the history list is
1010  *      empty. If the buffer is not empty, then the history list
1011  *      is not empty.
1012  */
1012
1013  /*
1014  *      If the buffer is empty, then the history list is
1015  *      empty. If the buffer is not empty, then the history list
1016  *      is not empty.
1017  */
1017
1018  /*
1019  *      If the buffer is empty, then the history list is
1020  *      empty. If the buffer is not empty, then the history list
1021  *      is not empty.
1022  */
1022
1023  /*
1024  *      If the buffer is empty, then the history list is
1025  *      empty. If the buffer is not empty, then the history list
1026  *      is not empty.
1027  */
1027
1028  /*
1029  *      If the buffer is empty, then the history list is
1030  *      empty. If the buffer is not empty, then the history list
1031  *      is not empty.
1032  */
1032
1033  /*
1034  *      If the buffer is empty, then the history list is
1035  *      empty. If the buffer is not empty, then the history list
1036  *      is not empty.
1037  */
1037
1038  /*
1039  *      If the buffer is empty, then the history list is
1040  *      empty. If the buffer is not empty, then the history list
1041  *      is not empty.
1042  */
1042
1043  /*
1044  *      If the buffer is empty, then the history list is
1045  *      empty. If the buffer is not empty, then the history list
1046  *      is not empty.
1047  */
1047
1048  /*
1049  *      If the buffer is empty, then the history list is
1050  *      empty. If the buffer is not empty, then the history list
1051  *      is not empty.
1052  */
1052
1053  /*
1054  *      If the buffer is empty, then the history list is
1055  *      empty. If the buffer is not empty, then the history list
1056  *      is not empty.
1057  */
1057
1058  /*
1059  *      If the buffer is empty, then the history list is
1060  *      empty. If the buffer is not empty, then the history list
1061  *      is not empty.
1062  */
1062
1063  /*
1064  *      If the buffer is empty, then the history list is
1065  *      empty. If the buffer is not empty, then the history list
1066  *      is not empty.
1067  */
1067
1068  /*
1069  *      If the buffer is empty, then the history list is
1070  *      empty. If the buffer is not empty, then the history list
1071  *      is not empty.
1072  */
1072
1073  /*
1074  *      If the buffer is empty, then the history list is
1075  *      empty. If the buffer is not empty, then the history list
1076  *      is not empty.
1077  */
1077
1078  /*
1079  *      If the buffer is empty, then the history list is
1080  *      empty. If the buffer is not empty, then the history list
1081  *      is not empty.
1082  */
1082
1083  /*
1084  *      If the buffer is empty, then the history list is
1085  *      empty. If the buffer is not empty, then the history list
1086  *      is not empty.
1087  */
1087
1088  /*
1089  *      If the buffer is empty, then the history list is
1090  *      empty. If the buffer is not empty, then the history list
1091  *      is not empty.
1092  */
1092
1093  /*
1094  *      If the buffer is empty, then the history list is
1095  *      empty. If the buffer is not empty, then the history list
1096  *      is not empty.
1097  */
1097
1098  /*
1099  *      If the buffer is empty, then the history list is
1100  *      empty. If the buffer is not empty, then the history list
1101  *      is not empty.
1102  */
1102
1103  /*
1104  *      If the buffer is empty, then the history list is
1105  *      empty. If the buffer is not empty, then the history list
1106  *      is not empty.
1107  */
1107
1108  /*
1109  *      If the buffer is empty, then the history list is
1110  *      empty. If the buffer is not empty, then the history list
1111  *      is not empty.
1112  */
1112
1113  /*
1114  *      If the buffer is empty, then the history list is
1115  *      empty. If the buffer is not empty, then the history list
1116  *      is not empty.
1117  */
1117
1118  /*
1119  *      If the buffer is empty, then the history list is
1120  *      empty. If the buffer is not empty, then the history list
1121  *      is not empty.
1122  */
1122
1123  /*
1124  *      If the buffer is empty, then the history list is
1125  *      empty. If the buffer is not empty, then the history list
1126  *      is not empty.
1127  */
1127
1128  /*
1129  *      If the buffer is empty, then the history list is
1130  *      empty. If the buffer is not empty, then the history list
1131  *      is not empty.
1132  */
1132
1133  /*
1134  *      If the buffer is empty, then the history list is
1135  *      empty. If the buffer is not empty, then the history list
1136  *      is not empty.
1137  */
1137
1138  /*
1139  *      If the buffer is empty, then the history list is
1140  *      empty. If the buffer is not empty, then the history list
1141  *      is not empty.
1142  */
1142
1143  /*
1144  *      If the buffer is empty, then the history list is
1145  *      empty. If the buffer is not empty, then the history list
1146  *      is not empty.
1147  */
1147
1148  /*
1149  *      If the buffer is empty, then the history list is
1150  *      empty. If the buffer is not empty, then the history list
1151  *      is not empty.
1152  */
1152
1153  /*
1154  *      If the buffer is empty, then the history list is
1155  *      empty. If the buffer is not empty, then the history list
1156  *      is not empty.
1157  */
1157
1158  /*
1159  *      If the buffer is empty, then the history list is
1160  *      empty. If the buffer is not empty, then the history list
1161  *      is not empty.
1162  */
1162
1163  /*
1164  *      If the buffer is
```

# DO YOU KNOW XTREE?



*This interface helps thousands of computer users all over the world organize their files everyday.*

**D**o you know me? I'm the most powerful graphic file and directory management program money can buy. Thousands of people use me everyday because I make file and directory organization easy.

My commands are simple. Just one keystroke lets you access, delete, rename, view, move, list, or print any and all files within any and all directories on a floppy and hard disk. Still some people don't recognize me.

My clear graphic representation of your directory tree structure is recognized world over as making PC-DOS simple to understand—even for beginners. Yet I'm powerful enough for the most discriminating DOS users.

I'm also a time saver, work saver and file saver. With me, reorganizing or creating your directory structure takes only minutes, instead of hours. And you'll never, ever misplace a file again.

Software reviewers know me as one of the most useful programs ever developed for hard disk users. *PC WEEK* said my performance was "unparalleled". *PC Magazine* praised my power and "quick as lightening speed". The entire *InfoWorld* staff uses me and even John Dvorak recommends me "highly".

And at \$49.95, complete with a 30-day money back guarantee, my users know me as a real value. I'm XTREE, recognized worldwide as "The new standard for file and directory management."

**XTREE:**  
**Don't use DOS without it!**

**SYSTEM REQUIREMENTS:** IBM PC/AT/XT or compatibles including Tandy 2000 with 192K of memory, and MS-DOS 2.0 or PC-DOS. **Not copy-protected.**

**Yes!** I want to know XTREE at \$49.95 each, plus \$3.00 s/h (CA residents add 6% sales tax. International orders add \$10.00 s/h.)

Send me \_\_\_\_\_ copies. Total enclosed \_\_\_\_\_.  
Payment:  VISA  MasterCard  AmEx  Check

Credit Card Expiration Date: \_\_\_\_\_

Card # \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City / State / Zip \_\_\_\_\_

*To order or to find your nearest dealer contact:*

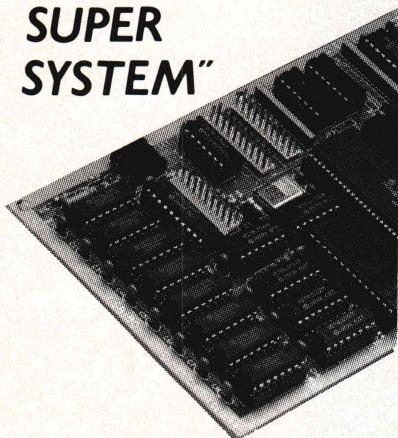
**Executive Systems Inc.**

15300 Ventura Blvd. Suite 305 • Sherman Oaks, CA 91403

**(800) 634-5545 or (800) 551-5353** in CA.

Byte Magazine called it.

# "CIARCA'S SUPER SYSTEM"



## The SB180 Computer/Controller

Featured on the cover of Byte, Sept. 1985, the SB180 lets CP/M users upgrade to a fast, 4" x 7 1/2" single board system.

- **6MHz 64180 CPU**  
(Z80 instruction superset), 256K RAM, 8K Monitor ROM with device test, disk format, read/write.
- **Mini/Micro Floppy Controller**  
(1-4 drives, Single/Double Density, 1-2 sided, 40/77/80 track 3 1/2", 5 1/4" and 8" drives).
- **Measures 4" x 7 1/2", with mounting holes**
- **One Centronics Printer Port**
- **Two RS232C Serial Ports**  
(75-19,200 baud with console port auto-baud rate select).
- **Power Supply Requirements**  
+5V +/- 5% @500 mA  
+12V +/- 20% @40mA
- **ZCPR3 (CP/M 2.2/3 compatible)**
- **Multiple disk formats supported**
- **Menu-based system customization**

### SB180-1

SB180 computer board w/256K bytes RAM and ROM monitor ..... \$369.00

### SB180-1-20

same as above w/ZCPR3, ZRDOS and BIOS source..... \$499.00

-Quantity discounts available-

### NEW

COMM180-M-S  
optional peripheral board adds  
1200 bps modem and SCSI/  
hard disk interface.

TO ORDER  
CALL TOLL FREE  
1-800-635-3355

TELEX  
643331

For technical assistance or  
to request a data sheet, call:

1-203-871-6170



Micromint, Inc.  
25 Terrace Drive  
Vernon, CT 06066

# C CHEST

## LISTING TWO (Listing continued, text begins on page 16)

```
329 }  
330 }  
331 }  
332 }  
333 }  
334 }  
335 }  
336 }  
337 }  
338 }  
339 }  
340 }  
341 }  
342 }  
343 }  
344 }  
345 }  
346 }  
347 }  
348 }  
349 }  
350 }  
351 }  
352 */-----*/  
353 #ifdef DEBUG  
354 main()  
355 {  
356 /* Test the history function. Everything typed at the  
357 * console is added to the history list and all of  
358 * the history expansion functions should work.  
359 */  
360 char buf[132];  
361 while(1)  
362 {  
363 printf("[%d] ", get_hnum());  
364 if( !gets(buf) )  
365 break;  
366 history( buf, 132 );  
367 if( *buf == 'h' && !buf[1] )  
368 print_hist( stdout, 0 );  
369 }  
370 }  
371 #endif
```

End Listing Two

## LISTING THREE

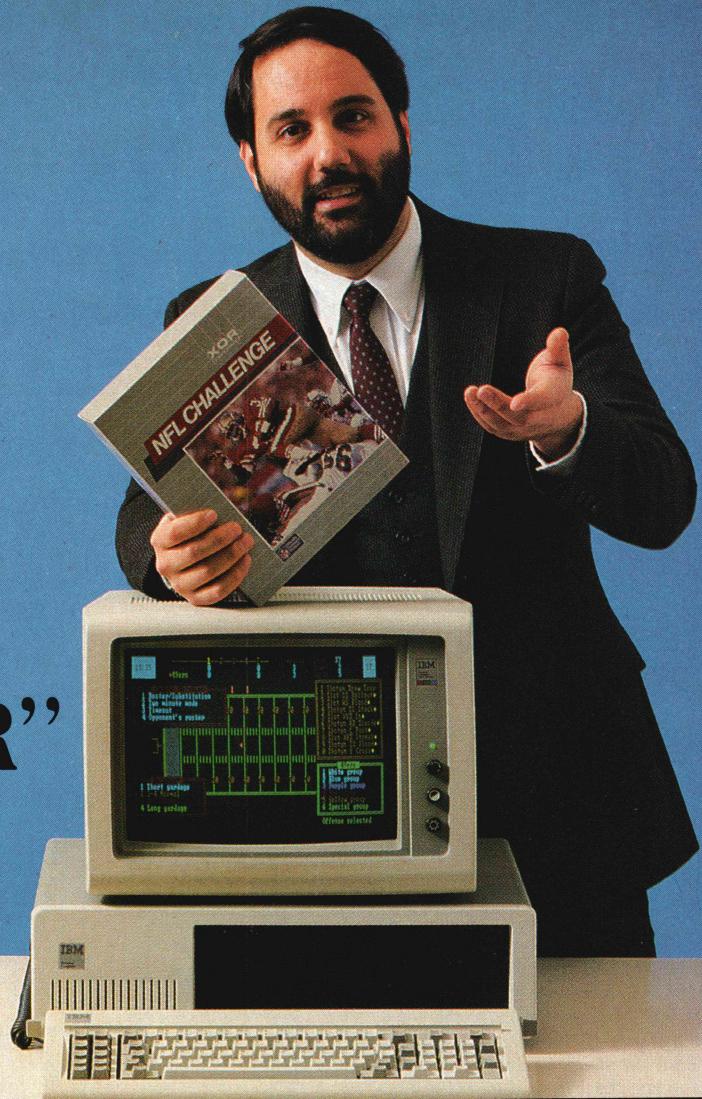
```
1 #include <stdio.h>  
2 #include <ctype.h>  
3 /* VAR.C Support for shell variables and aliases.  
4 * Copyright (c) 1985, Allen I. Holub. All rights reserved.  
5 * Externally accessible routines:  
6 */  
7 void unsetvar( name )  
8 int setvar( name, val )  
9 void printalias()  
10 void printvars()  
11 int getvar(srccp, destp, maxcount)  
12 /* void unsetvar( name ) Delete a variable or alias  
13 * int setvar( name, val ) Create/init a variable or alias  
14 * void printalias() Print all aliases  
15 * void printvars() Print all shell variables  
16 * int getvar(srccp, destp, maxcount) Expand a variable or alias  
17 * The same routines are used for both shell variables and aliases. The  
18 * latter have the high bit of the first character in the name set.  
19 */  
20 /* Isname is true if c is legal in a name. NAMELEN is the maximum  
21 * length of a name (additional characters are truncated).  
22 */  
23  
24 #define isname(c) (isalnum(c) || (c)=='-' || (c)=='-' || (c)==':')  
25 #define isalias(p) (*p & 0x80)  
26  
27 #define NAMELEN 8  
28  
29 typedef struct var  
30 {  
31 char name[NAMELEN];  
32 struct var *next;  
33 char val[1];  
34 }  
35 VAR;  
36  
37 static VAR *Varlist = NULL;  
38 static VAR *Lastvar = NULL;  
39 extern char *getenv( char* );  
40  
41 */-----*/  
42  
43 static VAR *findvar( name )  
44 char *name;  
45 {
```

(Continued on page 72)

# “The C86 C Compiler is Great...”

## Computer Innovations’ Support is Even GREATER”

DALE HILLMAN,  
PRESIDENT, XOR CORPORATION  
CREATOR OF “NFL CHALLENGE”



When Dale Hillman decided to create the most exciting football simulation game ever, he knew he needed good language support. The portability and maintainability of C made it a natural choice. **Which** C compiler to choose was another matter entirely.

“Of the many C compilers available, choosing the best one for the job was not easy. Comparing benchmarks, most compilers were strong in one or two categories, yet decidedly weak in others.

Computer Innovations’ C86 was the exception. I found the C86 Compiler consistently strong in all categories.

“C86 had a reputation for being a solid, reliable, high-performance compiler. 8087 math support, source level debugging — it had it all. BEST of all was Computer Innovations’ incredible technical support. Their highly knowledgeable support team was always available. Their assistance helped cut development

time substantially. And since NFL CHALLENGE took 12 1/2 man-years to create — every little bit helped. It was a service you just can’t place a dollar value on...”

If you’re working on the next great program, call Computer Innovations. We’ll show you why you’ll never have to look any further than C86.

**For Further Details  
Call Toll-Free:  
800-922-0169**

### Behind Innovative Programs — Computer Innovations

 COMPUTER  
INNOVATIONS, INC.

980 Shrewsbury Avenue,  
Tinton Falls, NJ 07724 USA (201) 542-5920

EUROPEAN DISTRIBUTOR  
Boston Micro, Inc., TELEX: 6712477 BMI USA

©1986 Computer Innovations, Inc.

\* NFL Challenge is a trademark of NFL Properties

# Structured Language World

Don't you need a publication that explores the structured language field—from new companies, to new applications for these powerful programming languages? If you are involved with languages including Ada, Pascal, Modula-2 and Algol, you owe it to yourself to subscribe to **Structured Language World**.

A current issue of the publication featured the following articles: the windowing capability enhancement for the UCSD p-system; SofTech's UCSD Pascal package for the IBM PC; Modula-2 versus Pascal, Ada-SynCh, Intermetrics' Ada Syntax Checker; PDOS Pascal designed to aid the writing of process control programs.

## Structured Language World

Title No. 368

1 year (4 issues): \$15.00

# A Guide to Modula-2

K. Christian

Modula-2 is a simple yet powerful programming language based on Pascal. It can be used in a wide variety of applications. This book is a complete guide to the language, organized by language features so that it easily serves as a reference as well as a basic text. Each part of the presentation of Modula-2 is built on the previous chapter and this systematic approach will be useful to a wide variety of readers. There are numerous examples throughout the book and the appendices provide several practical summaries of key language features.

1986/approx 464 pp/46 illus/hardcover  
\$34.00

ISBN: 0-387-96242-5

(Texts and Monographs in Computer Science)

For more information, write:  
Springer-Verlag New York, Inc.

Attn.: C. Deno  
175 Fifth Avenue  
New York, NY 10010

 Springer-Verlag

New York Berlin Heidelberg Tokyo

# C CHEST

## LISTING THREE (Listing continued, text begins on page 16)

```
46      /*      Do a search in the Varlist linked list for a
47      *      variable called "name." Return a pointer to it if
48      *      it exists or 0 if it doesn't.
49      */
50
51      register VAR      *p;
52
53      for( p = Varlist; p != NULL ; p = p->next )
54          if( !strcmp(name, p->name, NAMELEN) )
55              return p;
56
57      return (VAR *)0 ;
58 }
59
60 /*-----*/
61
62 void      unsetvar( name )
63 char      *name;
64 {
65     /*      If the variable called name exists, delete it.
66     */
67
68     register VAR      *p      = Varlist ;
69     register VAR      *lastp  = Varlist ;
70
71     if( !p || !(name & 0x7f) )
72         return;
73
74     for( p = Varlist; p != NULL ; )
75     {
76         if( !strcmp(name, p->name, NAMELEN) )
77             {
78                 /* If lastp == p then there's only one
79                 * node in the list.
80                 */
81
82                 if( lastp == p )
83                     Varlist = Lastvar = NULL;
84                 else
85                     {
86                         if( !(lastp->next = p->next) )
87                             Lastvar = lastp;
88                     }
89
90                 free( p );
91                 break;
92             }
93
94             lastp = p ;
95             p = p->next;
96     }
97 }
98
99 /*-----*/
100
101 varcpy ( dest , src )
102 register char      *dest, *src;
103 {
104     /*      Copy src to dest, stripping backslashes and quotes.
105     *      as appropriate (ie a \ within a quoted string isn't
106     *      stripped.
107     */
108
109     int      inquote = 0;
110
111     while( *src )
112     {
113         if( *src == '\\\'')
114         {
115             if( !inquote )
116                 src++;
117             else
118                 *dest++ = *src++;
119
120             if( *src )
121                 *dest++ = *src++;
122         }
123         else if( *src == '\"' || *src == '\'')
124         {
125             inquote = ~inquote;
126             src++;
127         }
128         else
129             *dest++ = *src++;
130     }
131
132     *dest = 0;
133 }
134
135 /*-----*/
136
137 int      setvar( name, val )
138 char      *name, *val;
139 {
140     /*      Set the value of the variable "name" to "val". If it
141     *      already exists, delete it.
142     */
143
144     register VAR      *vp;
145
146     if( !*name )
147         return 0;
148
149     unsetvar( name );
150
151     if( !(vp = (VAR *) malloc(sizeof(VAR) + strlen(val))) )
152         return 0;
153 }
```

Circle no. 236 on reader service card.

# Product Information

Free!

Postage Paid!

Free!

## Dr. Dobb's Journal of Software Tools

February 1986 #112

Expiration Date: May 30, 1986

Name \_\_\_\_\_ Title \_\_\_\_\_

Company \_\_\_\_\_ Phone \_\_\_\_\_

Address \_\_\_\_\_

City/State/Zip \_\_\_\_\_

Please circle one letter in each category:

**I. My work is performed:**

- A. for in-house use only.
- B. for other companies.
- C. for end users/retailers.
- D. in none of the above areas.

**II. My primary job function:**

- A. Software Project Mgmt/Sprv
- B. Hardware Project Mgmt/Sprv
- C. Computer Consultant
- D. Corporate Management
- E. Other

**III. My company department performs:**

- A. software development.
- B. computer system integration.
- C. computer manufacturing.
- D. computer consulting.
- E. computer research.
- F. none of the above.

**IV. This inquiry is for:**

- A. a purchase within 1 month.
- B. a purchase within 1 to 6 months.
- C. product information only.

**V. Corporate Purchase Authority:**

- A. Final Decision-maker
- B. Approve/Recommend
- C. No Influence

**VI. Personal Computer Users at my Jobsite:**

- A. 10,000 or more
- B. 500 to 9,999
- C. 100 to 499
- D. 10 to 99
- E. less than 10

**VII. On average, I advise others about computers:**

- A. more than once per day.
- B. once per day.
- C. once per week.
- D. less than once per week.

**VIII. In my job function, I:**

- A. design software and/or write code.
- B. design software.
- C. write code.
- D. don't design software or write code.

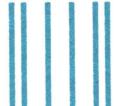
A Reader Service Number appears on each advertisement. Circle the corresponding numbers below for more info.

001	002	003	004	005	006	007	008	009
010	011	012	013	014	015	016	017	018
019	020	021	022	023	024	025	026	027
028	029	030	031	032	033	034	035	036
037	038	039	040	041	042	043	044	045
046	047	048	049	050	051	052	053	054
055	056	057	058	059	060	061	062	063
064	065	066	067	068	069	070	071	072
073	074	075	076	077	078	079	080	081
082	083	084	085	086	087	088	089	090
091	092	093	094	095	096	097	098	099
100	101	102	103	104	105	106	107	108
109	110	111	112	113	114	115	116	117
118	119	120	121	122	123	124	125	126
127	128	129	130	131	132	133	134	135
136	137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152	153
154	155	156	157	158	159	160	161	162
163	164	165	166	167	168	169	170	171
172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198
199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216
217	218	219	220	221	222	223	224	225
226	227	228	229	230	231	232	233	234
235	236	237	238	239	240	241	242	243
244	245	246	247	248	249	250	251	252
253	254	255	256	257	258	259	260	261
262	263	264	265	266	267	268	269	270
271	272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287	288
289	290	291	292	293	294	295	296	297
298	299	999						

Circle 999 to start a 12 month subscription at the price of \$29.97

Thank You!

Dr. Dobb's greatly appreciates your responses to questions I through VIII.



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

BUSINESS REPLY MAIL

First Class Permit #217, Clinton, Iowa

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of

**Software Tools**

P.O. Box 2157

Clinton, Iowa 52735-2157





NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

## BUSINESS REPLY MAIL

First Class Permit #217, Clinton, Iowa

POSTAGE WILL BE PAID BY ADDRESSEE

**Dr. Dobb's Journal of**

# Software Tools

P.O. Box 2157

Clinton, Iowa 52735-2157



**Free!**

**Postage Paid!**

**Thank You!**  
**Dr. Dobb's greatly appreciates your**  
**responses to questions I through VIII.**

A Reader Service number appears on each advertisement. Circle the corresponding numbers below for more info.

001 002 003 004 005 006 007 008 009  
 010 011 012 013 014 015 016 017 018  
 028 029 030 031 032 033 034 035 036  
 037 038 039 040 041 042 043 044 045  
 046 047 048 049 050 051 052 053 054  
 055 056 057 058 059 060 061 062 063  
 064 065 066 067 068 069 070 071 072  
 073 074 075 076 077 078 079 080 081  
 082 083 084 085 086 087 088 089 090  
 091 092 093 094 095 096 097 098 099  
 100 101 102 103 104 105 106 107 108  
 109 110 111 112 113 114 115 116 117  
 118 119 120 121 122 123 124 125 126  
 127 128 129 130 131 132 133 134 135  
 136 137 138 139 140 141 142 143 144  
 145 146 147 148 149 150 151 152 153  
 154 155 156 157 158 159 160 161 162  
 163 164 165 166 167 168 169 170 171  
 172 173 174 175 176 177 178 179 180  
 181 182 183 184 185 186 187 188 189  
 190 191 192 193 194 195 196 197 198  
 199 200 201 202 203 204 205 206 207  
 208 209 210 211 212 213 214 215 216  
 226 227 228 229 230 231 232 233 234  
 235 236 237 238 239 240 241 242 243  
 244 245 246 247 248 249 250 251 252  
 253 254 255 256 257 258 259 260 261  
 262 263 264 265 266 267 268 269 270  
 271 272 273 274 275 276 277 278 279  
 280 281 282 283 284 285 286 287 288  
 289 290 291 292 293 294 295 296 297  
 298 299 999

Circle 999 to start a 12 month subscription at the price of \$29.97

## Dr. Dobb's Journal of Software Tools

February 1986 #112

Expiration Date: May 30, 1986

Name \_\_\_\_\_ Title \_\_\_\_\_

Company \_\_\_\_\_ Phone \_\_\_\_\_

Address \_\_\_\_\_

City/State/Zip \_\_\_\_\_

Please circle one letter in each category:

**I. My work is performed:**

- A. for in-house use only.
- B. for other companies.
- C. for end users/retailers.
- D. in none of the above areas.

**II. My primary job function:**

- A. Software Project Mgmt/Sprv
- B. Hardware Project Mgmt/Sprv
- C. Computer Consultant
- D. Corporate Management
- E. Other

**III. My company department performs:**

- A. software development.
- B. computer system integration.
- C. computer manufacturing.
- D. computer consulting.
- E. computer research.
- F. none of the above.

**IV. This inquiry is for:**

- A. a purchase within 1 month.
- B. a purchase within 1 to 6 months.
- C. product information only.

**V. Corporate Purchase Authority:**

- A. Final Decision-maker
- B. Approve/Recommend
- C. No Influence

**VI. Personal Computer Users at my Jobsite:**

- A. 10,000 or more
- B. 500 to 9,999
- C. 100 to 499
- D. 10 to 99
- E. less than 10

**VII. On average, I advise others about computers:**

- A. more than once per day.
- B. once per day.
- C. once per week.
- D. less than once per week.

**VIII. In my job function, I:**

- A. design software and/or write code.
- B. design software.
- C. write code.
- D. don't design software or write code.

## Product Information

**Free!**

```

154 strcpy( vp->name, name, NAMELEN );
155 varcpy( vp->val, val );
156 vp->next = NULL;
157
158 if( !Varlist )
159     Varlist = vp;
160 else
161     Lastvar->next = vp;
162
163 Lastvar = vp;
164 return 1;
165 }
166
167 /-----
168 * All routines in this module may be used for both aliases and shell
169 * variables except for the two print routines. Alias's have
170 * the high bit if the first character of the name string set.
171 */
172
173 void printalias()
174 {
175     register VAR *p;
176
177     for( p = Varlist; p != NULL; p = p->next)
178         if( isalias(p->name) )
179             printf("%c%-8.8s: %s\n",
180                   *(p->name) & 0x7f, p->name +1, p->val );
181 }
182
183 void printvars()
184 {
185     register VAR *p;
186
187     for( p = Varlist; p != NULL; p = p->next)
188         if( !isalias(p->name) )
189             printf("%-8.8s: %s\n", p->name, p->val );
190 }
191
192 /-----
193
194 int getvar( srcp, destp, maxcountp )
195 char **destp, **srcp;
196 int *maxcountp;
197 {
198     /* Expand variable at "srcp" into destp. Do nothing if
199     * the variable doesn't exist, otherwise update srcp to
200     * point past the variable name, update destp to point
201     * past the end of the expanded variable, and decrement
202     * maxcountp by the proper amount.
203     *
204     * Both shell variables and environment variables will be
205     * expanded. However, shell variables take precedence
206     * over environments (they're looked for first).
207     *
208     * A name can consist of any character in the set
209     * {a-zA-Z0-9_}.
210     *
211     * If the name is for an alias it will be copied to dest if
212     * it can't be expanded, otherwise the name is discarded
213     * if it can't be expanded.
214     */
215
216     register VAR *vp;
217     char name[NAMELEN+1];
218     register char *p = name;
219     int i = NAMELEN;
220
221     /* If the source string is empty or doesn't contain a legit.
222     * name then return, doing nothing to dest.
223     */
224
225     if( !**srcp || !isname(**srcp & 0x7f) )
226         return 0;
227
228     /* Extract the name from the string at *srcp, updating
229     * *srcp to point past it.
230     */
231
232     for( **srcp; isname(**srcp & 0x7f) && --i>=0; *p++ = *(*srcp)++ )
233     ;
234
235     *p = 0;
236
237     /* Now look for the name. If findvar returns true it's
238     * a variable or alias.
239     */
240
241     if( vp = findvar(name) )
242     {
243         /* Expand the alias. Note that the since we
244         * are called from exp_vars, this next
245         * call is a second order recursion.
246         */
247
248         i = exp_vars( *destp, vp->val, *maxcountp, 2 );
249         *destp += i;
250         *maxcountp -= i;
251     }
252
253     else
254     {
255         /* It's not in the variable table. If we're processing
256         * an alias, copy the name to dest. If we're not
257         * processing an alias, but the variable is in the
258         * environment, then copy the contents of the environment
259         * else do nothing.
260         */
261
262         if( p = isalias(name) ? name : getenv(name) )
263             while( *maxcountp > 1 && *p )

```

(Continued on next page)



Stop Thinking-  
Start Programming Today!

SPECIAL INTRODUCTORY OFFER!  
C' Prime, Personal Computing and C,  
Plus Apprentice C.  
A \$169 value only **\$99**

NEW FROM MANX AZTEC!

**C' Prime** ~~\$99~~ **\$79**

Never has C been easier to learn. **Manx Aztec** is now offering a complete C system called **C' Prime** at an exceptionally low price. This powerful system includes a Compiler, Linker, Assembler, Editor, Libraries and Object Librarian. **C PRIME** supports a host of third-party software.

**C Apprentice** ~~\$49.95~~ **\$39.95**

Learn C quickly with this complete, easy-to-use C language interpreter. **Apprentice C** includes a complete one-step compiler that executes with lightning speed, an editor, and a run-time system.

NEW FROM ASHTON-TATE!

**Personal Computing and C**

A detailed, easy-to-understand guide to C programming prepared especially by Ashton-Tate for use with the new **Aztec C' Prime**. Includes chapters on C programming basics, function libraries, data handling, and advanced features, plus a complete money management demonstration program.

To order or for information call:

**TEC-WARE**  
1-800-TEC-WARE  
(In NJ call 201-530-6307)



UNIX is a registered TM of Bell Laboratories. ©1986 TM Ashton Tate, Inc., MANX AZTEC, C PRIME.  
Apprentice C, TM Manx Software Systems, Inc.

Circle no. 222 on reader service card.

# Put More UNIX™ in Your C.

**Unitools \$99**

**MAKE, DIFF and GREP**

These versatile UNIX-style utilities put power at your fingertips. MAKE, a program administrative tool, is like having an assistant programmer at your side. DIFF compares files and shows you the differences between them.

GREP can search one or many files looking for one pattern or a host of patterns.



**“Z” \$99**

**A Powerful “vi”-type Editor**

Similar to the Berkeley “vi” editor, “Z’s” commands are flexible, terse, and powerful; macro functions give you unlimited range. Features include “undo,” sophisticated search and replace functions, automatic indentation, C-tags, and much, much more.



**PC-LINT \$99**

**Error Checking Utility**

A LINT-like utility that analyzes programs and uncovers bugs, quirks and inconsistencies. Detects subtle errors. Supports large and small memory models, has clear error messages and executes quickly. Has lots of options and features that you wouldn’t expect at this low price.



**SunScreen \$99**

**Low-priced Screen Utility**

This versatile graphics package easily creates and modifies formatted screens, validates fields, supports function keys, color and monochrome cards. With library source SunScreen is \$199.



**Compatible with all leading MS/  
PC-DOS C compilers.**

## SPECIAL OFFER:

Unitools, “Z,”  
PC-LINT and Sun-  
Screen All for only

**\$349**

# C CHEST

## LISTING THREE (Listing continued, text begins on page 16)

```
264 {  
265     **destp = *p++;  
266     (*destp)++;  
267     (*maxCountp)--;  
268 }  
269 }  
270 }  
271  
272     **destp = '\0';  
273 }
```

**End Listing Three**

## LISTING FOUR

```
1 #include <stdio.h>  
2  
3 /*      UNARGV.C      Concatenate all argv entries into a single  
4  *      string.  
5  */  
6  
7 int    unargv( argc, argv, dest, maxcount )  
8 char   **argv, *dest;  
9 register int maxcount;  
10 {  
11     /* Turn argv into a single string, with a single ' ' separating  
12     * each entry. maxcount is the maximum size of dest. Return  
13     * the number of characters put into dest.  
14     */  
15  
16     register char  *src;  
17     char   *sdest = dest;  
18  
19     while( --argc >= 0 && maxcount > 0 )  
20     {  
21         for( src = *argv++; *src && --maxcount > 0; )  
22             *dest++ = *src++;  
23  
24         if( --maxcount > 0 && argc > 0 )  
25             *dest++ = ' ';  
26     }  
27  
28     *dest = 0;  
29     return( dest - sdest );  
30 }
```

**End Listing Four**

## LISTING FIVE

```
1 extern int    strcpy ( char*, char* );  
2 extern char   *malloc ( unsigned );  
3 extern int    strlen ( char* );  
4  
5 char   *strsave( str )  
6 char   *str;  
7 {  
8     /* Save the indicated string in a malloc()ed section  
9      * of static memory. Return a pointer to the copy or  
10     * 0 if malloc failed  
11     */  
12  
13     register char *rptr;  
14     extern char *malloc();  
15  
16     if( rptr = malloc( strlen(str) + 1 ) )  
17     {  
18         strcpy( rptr, str );  
19         return rptr;  
20     }  
21  
22     return (char *)0;  
23 }  
24 }
```

**End Listings**

To order or for information call:

**TECWARE**  
1-800-TEC-WARE

(In NJ call 201-530-6307)



Circle no. 223 on reader service card.

# WE'RE NOT LIKE OUR COMPETITION

• **YOU LIKE IT, OR WE TAKE IT BACK!** If for any reason, you are not satisfied with any product you purchase, you may return it within 10 days of receipt for replacement, credit or a full refund.\*

• **WE'LL PAY YOU IF YOU FIND A LOWER PRICE!** If you find a lower price in this issue before you buy, from any source that can deliver the identical product, we'll beat it! If you buy any item from us at pricing in this ad and find a lower price from any source in this issue, that can deliver the identical product, we'll not only refund the difference you paid, but also pay you 20% of the difference for your trouble!

• **GUARANTEED AVAILABILITY!** Any item you order will be shipped within Two working days (normally 12 hours) or you will be given a firm ship date when you order! If for any reason we cannot ship by the date you are given, we will deduct 5% from the price of the products shipped late and credit your order accordingly.

## FREE DIGITAL WATCH

With the purchase of any disk drive in this issue, we'll include a 7 melody alarm, Quartz chronograph, digital watch...absolutely FREE! (limit one per customer!)



## BORLAND BLOW OUT!

- SIDEKICK 1.5 c/p . . . . . \$27.75
- SIDEKICK 1.5 nc/p . . . . . \$42.99
- SIDEKICK MAC nc/p . . . . . \$42.99
- TURBO PASCAL 3.0 nc/p \$35.33
- SUPERKEY 1.1 nc/p . . . . . \$35.33

## NEW!

- TRAVELING SIDEKICK nc/p . . . . . \$39.97
- TURBO LIGHTNING nc/p . . . . . \$55.00
- REFLEX nc/p . . . . . \$55.00

## COMPONENTS

Quality Japanese mfg. from companies like HITACHI, TOSHIBA and FUJITSU.

- **256K DRAMS** Set of 9 150ns . . . . . \$26.46
- **64K DRAMS** Set of 9 150ns . . . . . \$7.56
- **8087-3** . . . . . \$99.00
- **8087-2** . . . . . \$129.60
- **80287** . . . . . \$178.00
- **27128** . . . . . \$2.90
- **70128** replaces 8088 . . . . . \$14.75
- **27256** . . . . . \$4.50
- **2764** . . . . . \$1.98
- **4128** . . . . . \$2.97

## NOVATION SMARTCAT PLUS

Auto answer/Auto dial 1200 & 0-300 bps. Hayes™ (AT) compatible. Includes MITE™ communication software.

Internal or external model . . . . . \$308.25

## ACCESSORIES

### KEYBOARDS

FULLY IBM™ and KEYTRONICS™ COMPATIBLE

- 5150 style . . . . . \$78
- 5151 style . . . . . \$98
- REPLACEMENT HARD DISK DRIVES
- 13MB ½ ht. . . . . \$296.00
- 25MB ½ ht. . . . . \$429.00
- 38MB Full ht. Seagate voice coil . . . . . \$833.00
- 51MB Full ht. Seagate voice coil . . . . . \$1044.00

### BOARD PRODUCTS

- Western Digital PC/AT type Hard/Floppy cont. . . . . \$297.00
- MULTITECH 2 Drive PC floppy controller . . . . . \$45.85
- MULTITECH 4 drive PC floppy controller . . . . . \$8.50
- MULTITECH multi. board (AST sixpack comp.) . . . . . \$119.45
- AST sixpack . . . . . \$223.00
- AST Advantage 128K . . . . . \$384.00
- MULTITECH color board . . . . . \$98.87
- HERCULES color board . . . . . \$144.00
- HERCULES graphics board . . . . . \$287.00
- QUADRAM Quadboard w/64K . . . . . \$197.00
- MULTITECH 384K mem. exp. board (empty) . . . . . \$56.00

### MODEMS

- SMARTTEAM 103/212A Hayes™ comp. . . . . \$184.00
- NOVATION 2400 Professional w/o software . . . . . \$498.00
- NOVATION 2400 with MS-DOS or Macintosh software . . . . . \$548.00
- HAYES 1200B w/Smartcom II . . . . . \$349.00

### MONITORS

- TAXAN 400 medium resolution RGB . . . . . \$253.00
- AMDEK 300G 12" Green . . . . . \$118.96
- TATUNG 12" Hi Resolution Amber . . . . . \$119.75
- TATUNG 14" Hi Resolution RGB color . . . . . \$444.50

### PC POWER SUPPLIES

- 150 WATT . . . . . \$99.00

## FLOPPY DRIVES

PC COMPATIBLE

- PANASONIC ½ Ht . . . . . \$88

- APPLE II Compatible, inc. cable . . . . . \$97.75

## HARD DRIVES

### COMPLETE INTERNAL SYSTEMS

Includes drive, controller card, cables and install procedures. Capacities listed are unformatted. We sell only the finest drives from Seagate, Mitsubishi, and others guaranteed to meet or exceed original manufacturer's specifications.

- 13MB ½ Ht. . . . . \$369
- 25MB ½ Ht. . . . . \$469
- 38MB Full Ht. SEAGATE Voice Coil . . . . . \$899
- 51MB Full Ht. SEAGATE Voice Coil . . . . . \$1098

## FUJI DISK EXPLOSION

Certified Quality 5 1/4" Bulk Disks manufactured by FUJI. Available in your choice of packaging.

In Perfect Data™ Dial-N-File box.

With sleeves, labels and w/p tabs. Priced per box of 10.

TYPE	1-19	20-99	100-499	500-999	1000+
SS/DD	10.75	9.97	9.57	9.23	8.99
DS/DD	13.50	12.67	11.97	11.45	10.99

In White Box

With sleeves, labels and w/p tabs. Priced per box of 10.

TYPE	1-19	20-99	100-499	500-999	1000+
DS/DD	10.99	10.37	9.97	9.60	9.26

DS/DD Bulk w/o box, sleeves, labels or w/p tabs

Price each, in increments of 50 only, poly bagged.

50	100-450	500-950	1000-4950	5000+
.96	.89	.84	.80	.77

Inside California

1-800-358-8881

Outside California

1-800-826-3736

### \* THE FINE PRINT!

Excluding software. Prior authorization required, all items returned must be in original condition with carton, packing and all manuals, etc. We accept Money Orders, Certified and Cashiers checks, personal checks (product shipped when check clears), VISA and MasterCard with no surcharge and American Express. All items in stock at time of publication and subject to prior sale at time of publication. All products shipped UPS ground unless specified otherwise. All normal manufacturer's warranties apply.

## WORLDWIDE ACCESS

Everybody hates us but our customers.

### IF YOU DON'T SEE IT, CALL!

We have virtually any product available at the best pricing. Space limits us to only a fraction of what we sell. Call us for a quote and delivery information. If we don't have it, we'll get it for you!

## LISTING ONE (Text begins on page 26)

```

{ file polydiv.pas, 85/9/18/tfr (from 9/15,13) }
{ simulation of CRC-type operations }

{ Copyright (c) 1985, T.F. Ritter; All Rights Reserved }

{ generates a binary trace through the polynomial division
{ and crc algorithms to illustrate equivalent results
{ (this assumes we init the remainder reg. to 0) }

{$R+} { Range Checks ON }

PROGRAM polydiv;

{ first, output through MSDOS, and define binary display }

TYPE
  regs = RECORD
    ax,bx,cx,dx,bp,si,di,ds,es,flags: INTEGER;
  END;

PROCEDURE bdosch( ch: CHAR );
{ output through MSDOS; allow Ctrl-P toggle }
  VAR r: regs;
  BEGIN
    r.ax := $0200;
    r.dx := ORD(ch);
    Msdos( r );
  END;

TYPE
  small = 0..15;

PROCEDURE showbin( x: INTEGER; from, too: small );
{ display subset of integer as binary bits }
{ from and too are place values (15 - 0) left to right }
  VAR i: small;
  BEGIN
    WRITE( ' ' );
    FOR i := 15 DOWNTO 0 DO
      BEGIN
        IF i IN [too..from] THEN
          IF (x < 0) THEN
            WRITE( ' 1' )
          ELSE
            WRITE( ' 0' )
        ELSE
          WRITE( ' x' );
        x := x Shl 1;
      END;
    WRITE( ' ' );
  END; {showbin}

VAR { globals }
  a: INTEGER; { the remainder value register; right-aligned }
  p: INTEGER; { the polynomial; right-aligned }
  d: INTEGER; { the data; left-aligned }
  deg: small; { the degree of the polynomial }
  dbits: BYTE; { the number of data bits to process }

PROCEDURE showad( pb: small );
{ show current remainder and next data bit (only) }
{ the data bit on the last step is meaningless }
  BEGIN
    WRITELN;
    showbin( a, (pb - 1), 0 );
    showbin( d, 15, 15 );
  END;

{ start bit-level utilities for crc-type algorithms }

TYPE
  abit = 0..1;

FUNCTION dn: abit;
{ value of next data bit }
{ since d is an INTEGER, "d Shl 1" = "d + d" }
{ use "d := d + d" in other Pascals }
  BEGIN
    IF (d < 0) THEN
      dn := 1
    ELSE
      dn := 0;
    d := d Shl 1;
  END;

FUNCTION an( n: small ): BOOLEAN;
{ value of particular remainder bit }
{ for other Pascals, use a loop and do "x Shl 1" n times }

{ "x Shl 1" must itself be "x + x"; init x to 1 first }
BEGIN
  an := ((a AND (1 SHL n)) <> 0);
END;

{ start crc-type algorithms }

PROCEDURE pdiv;
{ mod-2 polynomial division (for remainder only) }
  BEGIN
    IF (an(deg - 1) THEN
      a := (a Shl 1) XOR p XOR dn
    ELSE
      a := (a Shl 1) XOR dn;
  END; {pdv}

PROCEDURE crc;
{ mod-2 remainder without extra zeros }
  BEGIN
    IF (an(deg - 1) XOR (dn <> 0)) THEN
      a := (a Shl 1) XOR p
    ELSE
      a := (a Shl 1);
  END; {crc}

{ start trace displays; show one or the other }

PROCEDURE showdiv( dbits: BYTE );
  VAR i: BYTE;
  BEGIN
    WRITE( '^M^J'POLYNOMIAL DIVIDE; Polynomial = ' );
    showbin( p, deg, 0 );
    WRITE( '^M^J'Remainder Data' );
    showad( deg );
    FOR i := 1 TO dbits DO
      BEGIN
        pdv;
        showad( deg );
      END;
    WRITELN;
  END;

PROCEDURE showcrc( dbits: BYTE );
  VAR i: BYTE;
  BEGIN
    WRITE( '^M^J'CRC OPERATION; Polynomial = ' );
    showbin( p, deg, 0 );
    WRITE( '^M^J'Remainder Data' );
    showad( deg );
    FOR i := 1 TO dbits DO
      BEGIN
        crc;
        showad( deg );
      END;
    WRITELN;
  END;

PROCEDURE init( i: BYTE );
{ select one of the initializations }

PROCEDURE init1;
{ simulating long division example }
{ from Peterson & Brown, Proc. of the IRE, Jan. 1961, p. 232 }
  BEGIN
    p := $0005;
    deg := 2;
    d := $e800;
    dbits := 6;
  END; {init1}

PROCEDURE init2;
{ simulating generation of check code }
{ Peterson & Brown, 1961, p. 229 }
{ data are shifted left, so are in reversed order from article }
  BEGIN
    p := $0035; { the classical example polynomial }
    deg := 5;
    d := $8940;
    dbits := 10;
  END; {init2}

PROCEDURE init3;
{ simulating published tables }
{ K. Rallapalli, EDN, Sept. 5 1978, pp. 119 - 123 }
  BEGIN
    p := $0035; { here it is again }
    deg := 5;
    d := $1b0;
    dbits := 12;
  END; {init3}

BEGIN {init}
CASE i OF
  1: init1;
  2: init2;
END;

```

(Continued on page 78)

THE 11 TH WEST COAST  
**COMPUTER FAIRE**

# Decade II

the odyssey continues . . .

APRIL 3-6, 1986  
MOSCONC CENTER  
SAN FRANCISCO

## Don't miss the best show of '86!

The biggest, most important, most exciting, public computer event on the West Coast will take place at the **11th West Coast Computer Faire**. People who are vitally interested in computers will be there. Join us at the show that makes the difference!

**See thousands** of computer products exhibited by hundreds of leading and innovative companies.

**Touch the future** with hands-on demonstrations.

**Learn from the experts.** The comprehensive conference program attracts industry speakers and computer users from business, education, government and scientific, engineering and research communities.

**Take advantage** of low cost Professional Development Seminars.

**Get FREE** one-on-one consulting help.

**Save money** with incredible "special show pricing" on all kinds of products.

### Send in for more information:

Please send me information on attendee preregistration discounts.

Name \_\_\_\_\_

Company \_\_\_\_\_

Street \_\_\_\_\_

City/Town \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Phone \_\_\_\_\_

I would like to  Attend  Exhibit

Send this coupon to:

**The 11th West Coast Computer Faire**  
181 Wells Avenue  
Newton, MA 02159

You can call us at (617) 965-8350.

DD

An exclusive production of **Computer Faire, Inc.** / A Prentice-Hall Company

Now! Automatic time and  
date stamping for  
CP/M® 2.2

# DATESTAMPER™

Avoid  
erasing the  
wrong files!

Back up  
files by time  
and date!

"DateStamper...  
is a real winner."

Bruce Morgen, Users  
Guide, Jul-Aug. 1985

Write or call for  
further information  
(714) 659-4432

Circle no. 193 on reader service card.

- Extends CP/M 2.2 to automatically record date and time a file is created, read and modified. DateStamper reads the exact time from a real-time clock if you have one, or records the order in which you use files each day.
- Date stamping information is contained in a separate file and read by our directory utility, SDD.COM. Powerful DATSWEET file management utility also included.
- Simple menu-driven installation. Large library of clocks supplied, with provision to add others. Many options configurable to your individual requirements.
- Disks prepared for date stamping are fully compatible with standard CP/M. DateStamper also runs with all versions of ZCPR, Z-RDOS and many other CP/M-80 modifications.

CP/M is a registered trademark of Digital Research, Inc.

Plu'Perfect Systems

8" SSSD, Kaypro, Osborne,  
H/Z-89 formats ..... \$49  
(Most other formats, add \$5) ..... \$3  
Shipping & handling ..... \$3  
California residents add 6% sales tax  
MasterCard &  
Visa accepted

BOX 1494, IDYLLWILD, CA 92349

## BOOKSHELF™

Fast, compact, high quality,  
easy-to-use CP/M system

Series 100

Priced from  
\$895.00

10MB System  
Only \$1645.00



### Features

- Ready-to-use professional CP/M computer system
- Works with any RS232C ASCII terminal (not included)
- Network available
- Compact 7.3 x 6.5 x 10.5 inches, 12.5 pounds, all-metal construction
- Powerful and Versatile:
  - Based on Little Board/PLUS single-board computer
  - Two RS232 serial ports
  - One Centronics printer port
  - One or two 400 or 800 KB floppy drives
  - 10-MB internal hard disk drive option

- Comprehensive Software Included:
  - Enhanced CP/M operating system with ZCPR3
  - Word processing, spreadsheet, relational database, spelling checker, and data encrypt/decrypt (T/MAKER III)
  - Operator-friendly shells; Menu, Friendly™
  - Read/write and format dozens of floppy formats (IBM PC-DOS, KAYPRO, OSBORNE, MORROW...)
  - Menu-based system customization
- Expandable:
  - Floppy expansion to four drives
  - Hard disk expansion to 60 megabytes
  - SCSI/PLUS™ multi-master I/O expansion bus

T/MAKER III is a trademark of T/Maker Company.  
IBM is a registered trademark of International Business Machines.  
Z80A is a registered trademark of Zilog, Inc.  
CP/M is a registered trademark of Digital Research.

**AMPRO**  
COMPUTERS, INCORPORATED

67 East Evelyn Ave. • Mountain View, CA 94041 • (415) 962-0230 • TELEX 4940302

Circle no. 158 on reader service card.

# CRC

## LISTING ONE

(Listings continued, text begins on page 26)

```

3: init3;
END; (case)
a := 0;
END; (init)

{ start of the ultimate command code }

VAR
  i: BYTE;

BEGIN (main)

ConOutPtr := Ofa( bdosch ); {allow Ctrl-P printer toggle}

{ for three different initializations . . . }
FOR i := 1 TO 3 DO
  { show binary trace of pdiv and crc algorithms }
  BEGIN
    init( i );
    showdiv( dbits + deg );
    init( i );
    showcrc( dbits );
  END;

END. (main)

{ end file polydiv }

```

End Listing One

## LISTING TWO

```

{ file crctime.pas, 85/9/18/tfr (from 9/15, 7/13, 6/27-26,
showtime.pas 3/23, 1/4, 84/12/25
and showkeys 12/12-11, 11/17
and strinkey.pas, 11/17) }

{ compute crc execution times }

{ Copyright (c) 1984, 1985, T.F. Ritter; All Rights Reserved }

{ Please feel free to copy and use the CRC routines, but . . .
{ do include the name of the CRC programmer when you do. }
{ The CRC programmer is Terry Ritter. }


```

```

{ AIMS:
{ 1. Display a number of different CRC-CCITT implementations.
{ 2. Verify identical results.
{ 3. Collect and display time statistics on each routine. }


```

PROGRAM showtime;

```

{ minimum overhead for speed tests }
{ SR- } { Range Checks OFF }
{ $C- } { Ctrl-C Checking OFF }


```

PROCEDURE ownership;

```

BEGIN
  WRITE( con,
  ^M^J, 'CRCTIME, 85/9/18',
  ^M^J, 'Execution times for various CRC implementations.',
  ^M^J, 'Copyright (c) 1985, T.F. Ritter; All Rights Reserved.',
  ^M^J );
END;


```

TYPE

```

regs = RECORD
  ax,bx,cx,dx,bp,si,di,ds,es,flags: INTEGER;
END;


```

```

PROCEDURE bdosch( ch: CHAR );
{ direct console output through MSDOS }
{ allows Ctrl-P printer toggle }
{ also skips Ctrl-C break test }


```

VAR

```

  r: regs;
BEGIN
  r.ax := $0200;
  r.dx := ORD(ch);
  MsDos( r );
END;


```

{ start of time operations }

```

TYPE
  timearty = ARRAY[0..3] of INTEGER; {originally had yr, dy}


```

```

PROCEDURE readhms( VAR dt: timearty );
{ Hi(dt[2]) = hrs (0 - 23), Lo(dt[2]) = mins (0 - 59) }
{ Hi(dt[3]) = secs (0 - 59), Lo(dt[3]) = msecs (0 - 99) }


```

VAR

```

  r: regs;
BEGIN
  r.ax := $2c00;
  MsDos( r );
  dt[2] := r.cx;
  dt[3] := r.dx;
END; { rddti }


```

(Continued on page 80)

Includes complete source  
code and documentation!  
Only \$29.95 Each!

# A Unix-like Shell AND a Unix-like Utility Package

by Allen Holub

Both Available on Disk for MS-DOS

## THE SHELL

### SH—A Unix-like Shell for MS-DOS

*SH is an implementation of the most often used parts of the Unix C shell. This package includes an executable version of the shell along with the complete source code and full documentation.*

*Supported features are:*

**Editing** Command line editing with the cursors is supported. The line is visible as you edit it.

**Aliases** Can be used to change the names of commands or as very fast memory resident batch files.

**History** The ability to execute a previous command again. The command can be edited before being executed.

**Shell variables** Macros that can be used on the command line.

**Nested batch files** A batch file can call another batch file like a subroutine. Control is passed to the second file and then back to the first one when the second file is finished. DOS doesn't have this capability.

**Unix-like syntax** Slash (/) used as a directory separator, minus (-) as a switch designator. A 2048 byte command line is supported.

*The shell also supports redirection of standard input, standard output, and standard error.*

*This version corrects several bugs found in the original version printed in Dr. Dobb's Journal, December 1985 through March 1986 issues. It runs on any MS-DOS computer.*

## /util

### A Unix-like Utility Package for MS-DOS

*/util is a collection of Unix utility programs for MS-DOS. This package includes complete source code. All programs (and most of the utility subroutines) are fully documented. You'll find executable versions of:*

**cat** A file concatenation and viewing program

**cp** A file copy utility

**date** Prints the current time and date

**du** Prints amount of space available and used on a disk

**echo** Echoes its arguments to standard output

**grep** Searches for a pattern defined by a regular expression.

**ls** Gets a sorted directory.

**mkdir** Creates a directory

**mv** Renames a file or directory. Moves files to another directory.

**p** Prints a file, one page at a time.

**pause** Prints a message and waits for a response.

**printenv** Prints all the environment variables.

**rm** Deletes one or more files.

**rmdir** Deletes one or more directories.

**sub** Text substitution utility. Replaces all matches of a regular expression with another string.

To order, return this coupon with payment to: M&T Publishing, Inc., 2464 Embarcadero Way, Palo Alto, CA 94303

Please send me \_\_\_\_\_ copies of The Shell for \$29.95 each \_\_\_\_\_

\_\_\_\_\_ copies of /util for \$29.95 each \_\_\_\_\_

SUBTOTAL \_\_\_\_\_

CA residents add applicable sales tax \_\_\_\_\_ % \_\_\_\_\_

Please add \$1.75 per disk for shipping \_\_\_\_\_

TOTAL \_\_\_\_\_

**YES!**



Check/Money Order enclosed

Please charge my \_\_\_\_\_ VISA \_\_\_\_\_ M/C \_\_\_\_\_

Amer. Exp. \_\_\_\_\_

Exp. Date \_\_\_\_\_

Card # \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_

Zip \_\_\_\_\_

Please allow 2 weeks for delivery

3112H

# d/MULTI MULTIUSER dBASE for TurboDOS

TRUE File and Record Locking as easy as d-BASE-II. Unlimited users can perform the magic of dBASE in the program or interactive mode

- \* TurboDOS 1.3 or 1.4
- \* No Peeks or Pokes
- \* System Date and Time Functions
- \* Printspooler Controls up to 16 printers

## Martian Technologies . . .

**.CREATEing Multi-users from  
Single-users around the world**

### CALL FOR DETAILS

**Martian Technologies**

8348 Center Dr., Ste. F, La Mesa, CA 92041  
**(619) 464-2924**



Circle no. 189 on reader service card.

## Modula-2 for only \$150

Mac  
and  
PC



Modula-2, the brain child of Niklaus Wirth, is fast becoming the next major programming language. This state-of-the-art language is now available for your Apple Macintosh or IBM PC computer.

### Modula-2 advantages:

- Structured programming language
- Separate compilation with strong type checking
- Module version checking
- Optional run time range checking
- Eliminates many Pascal deficiencies

### Price includes:

- Full Wirth Modula-2 compiler
- Library modules for I/O and math routines
- Software development environment and tools
- Extensive documentation with index and numerous sample programs

No developer fees. Disks not copy-protected.

To order, call toll-free:

**1-800-545-4842**

Have charge card ready.

**NOW SHIPPING:**  
Wirth's One Pass  
Native Code PC Compiler  
and  
Mac Modula-2 V4.0

**MODULA**  
CORPORATION

Modula Corporation  
950 North University Avenue  
Provo, Utah 84604  
(801) 375-7400

© 1986 Modula Corporation. Allow 4-6 weeks for delivery.

Circle no. 238 on reader service card.

## CRC

### LISTING TWO

(Listings continued, text begins on page 26)

```
FUNCTION timetorealsecs( x: timearty ): REAL;
BEGIN
  timetorealsecs := (Hi(x[2]) * 3600.0) + (Lo(x[2]) * 60.0) +
    Hi(x[3]) + (Lo(x[3]) / 100.0);
END;
```

```
FUNCTION timedif( a, b: timearty ): REAL;
BEGIN
  timedif := timetorealsecs( b ) - timetorealsecs( a );
END;
```

```
{ start of CRC stuff
{ byte crc in Turbo Pascal
{ for MSDOS }
```

```
{ METHOD 1: Pascal Bit-By-Bit
{ bit manipulation in a high-level-language
{ simulation of "longhand" division
{ mostly for verification of other methods }
```

```
PROCEDURE crcittb( VAR a: INTEGER; databit: BOOLEAN );
{ single bit computation; simulate polydiv hardware }
{ other Pascals can replace "a SHL 1" with "a + a" }
BEGIN
  IF databit = (a < 0) THEN
    a := a Shl 1
  ELSE
    a := (a Shl 1) XOR $1021; { CRC-CCITT }
END; { crcittb }
```

```
PROCEDURE crcittby( VAR aa: INTEGER; d: INTEGER );
{ whole byte computation }
VAR
  i: INTEGER;
BEGIN
  d := d Shl 7;
  FOR i := 7 DOWNTO 0 DO
    BEGIN
      d := d Shl 1;
      crcittb( aa, (d < 0) );
      END;
END; { crcittby }
```

```
{ METHOD 2: Pascal Fast Bit-By-Bit
{ eliminates procedure-call overhead in the loop
{ similar to most XMODEM crc implementations }
```

```
PROCEDURE crcfbpb( VAR aa: INTEGER; d: INTEGER );
{ fast bit-by-bit whole byte computation }
VAR
  i: INTEGER;
BEGIN
  d := d Shl 7;
  FOR i := 7 DOWNTO 0 DO
    BEGIN
      d := d Shl 1;
      IF ((d XOR aa) < 0) THEN
        aa := (aa Shl 1) XOR $1021
      ELSE
        aa := aa Shl 1;
      END;
END; { crcfbpb }
```

```
{ METHOD 3: Pascal Byte
{ process the data byte without loops
{ transportable within Turbo Pascal, and fairly fast
{ may be slower in Pascals without bit-shifting operations }
```

```
PROCEDURE crcitt( VAR dx: INTEGER; data: INTEGER );
{ dx := crcTransform( dx; data ) }
{ polynomial = $11021 }

{ FOR OTHER PASCALS: }
{   Generate a Swap function "swap( x: INTEGER ): INTEGER" }
{   probably "swap := (x * 256) + (x DIV 256) would work" }
{   Lo(x) = x AND $00ff - x AND 255 }
{   x Shr 4 = x DIV 16 }
{   x Shl 4 = x * 16 }
{   x Shl 5 = x * 32 }

BEGIN { crcitt }
  dx := Swap( dx ) XOR data;
  dx := dx XOR ( Lo(dx) Shr 4 );
  dx := dx XOR ( Swap(Lo(dx)) Shl 4 ) XOR ( Lo(dx) Shl 5 );
END; { crcitt }
```

```

{ METHOD 4: Pascal Table }
{ pull changes from table, indexed by composite value }
{ still faster, but requires the table, and filling the table }
{ may be even faster, relatively, in another Pascal }
{ a slower routine is fine to fill the table }

VAR
  crctab: ARRAY[0..255] of INTEGER;

PROCEDURE crctabu( VAR crcreg: INTEGER; data: INTEGER );
  BEGIN
  crcreg := Swap(crcreg) XOR data;
  crcreg := (crcreg AND $ff00) XOR crctab[ Lo(crcreg) ];
  END;

PROCEDURE initctab;
  { use method 3 to init the table }
  VAR
    i: INTEGER;
  BEGIN
  FOR i := 0 TO 255 DO
    BEGIN
    crctab[i] := 0;
    crctab(i, i);
    END;
  END; { initctab}

{ METHOD 5: Machine Code Byte }
{ method 3 in "Inline" machine code (MSDOS) }
{ typically hidden away in an "Include" file }
{ other Pascals may need to modify the stack interface }

PROCEDURE mrcitt1( VAR dx: INTEGER; data: INTEGER );
  { a := crcTransform( dx; data ) }
  { for MSDOS }
  { polynomial = $11021 }

BEGIN { mrcitt1 }
  INLINE (
    $c4/$7e/<dx/ { es:di := [bp + "dx"] }
    $26/$8b/$05/ { ax := [es:di] }

    { dx := SWAP(dx) XOR data; }
    $86/$e0/ { ah <=> al }
    $33/$46/<data/ { ax := ax XOR [bp + "data"] }
    $89/$c2/ { dx := ax }

    { dx := dx XOR ( LO(dx) SHR 4 ); }
    $d0/$e8/ { al := al SHR 1 }
    $32/$d0/ { dl := dl XOR al }

    { dx := dx XOR ( ( LO(dx) ) SHL 4 ); }
    $88/$d4/ { ah := dl }
    $d0/$e4/ { ah := ah SHL 1 }
    $32/$f4/ { dh := dh XOR ah }

    { dx := dx XOR ( ( LO(dx) ) SHL 5 ); }
    $88/$d0/ { al := dl }
    $b4/$00/ { ah := 0 }
    $d1/$e0/ { ax := ax SHL 1 }
    $33/$d0/ { dx := dx XOR ax }

    $26/$89/$15 { es:[di] := dx } );
  END; { mrcitt1 }

{ METHOD 6: Machine Code Table }
{ here the stack parameter interface becomes significant }
{ note the exchange notation "x ::= y" in the comments }

PROCEDURE mrcitt3( VAR dx: INTEGER; data: INTEGER );
  { a := crcTransform( dx; data ) }
  { for MSDOS }
  { polynomial = $11021 }

BEGIN { mrcitt3 }
  INLINE (
    $c4/$7e/<dx/ { es:di := [bp + "dx"] }
    $26/$8b/$15/ { dx := [es:di] }

    { dx := Swap(dx) XOR data; }
    $86/$d6/ { dh := dl }
    $33/$56/<data/ { dx := dx XOR [bp + "data"] }

    { bx := Lo(dx) SHL 1; dl := 0; }
    $31/$db/ { bx := bx XOR bx }
    $86/$d3/ { bl := dl }
    $d1/$e3/ { bx := bx SHL 1 }

    { dx := dx XOR crctab[ bx ]; }
    $33/$97/>crctab/ { dx := dx XOR [bx + "crctab"] }

    $26/$89/$15 { [es:di] := dx } );
  END; { mrcitt3 }

```

(Continued on next page)

## Productivity Tools from SCE

• **The Seidl Make Utility (SMK)** is the most powerful make utility you can buy for MS-DOS. When changes are made to any program module, SMK will issue only those commands *necessary and sufficient* to rebuild the system. SMK uses a super fast, proprietary dependency analysis algorithm that analyzes *all* dependencies before rebuilding any files. SMK understands complicated dependencies involving nested include files, source, and object code libraries. A high-level dependency definition language makes setting up dependencies easy. It supports parameterized macros, local variables, for loops, constants, include files, command line parameters, in-line and block comments, full pathname and directory support, and more! Works with most compilers, assemblers, and linkers. **\$99.95**

• **The Seidl Version Manager (SVM)** is a state of the art version control system for MS-DOS. It maintains a complete revision history of any text file, without duplication, and can rebuild any previous version of the file. SVM has highly optimized compression routines included for large projects and archiving. Other features include: full pathname and directory support, wildcards, exception cases, on line help, typeset documentation, tutorial, automatic error recovery, and much more. SVM is ideal for all software and text development projects. No other version management system delivers the features, performance, and reliability of SVM. **\$299.95 (Available in March)**

• **SMK+SVM** together form an extremely powerful, fully integrated set of productivity tools. **\$379.95**

## SEIDL COMPUTER ENGINEERING

1163 E. Ogden Ave., Suite 705-171  
Naperville, IL 60540  
(312) 983-5477

Circle no. 114 on reader service card.

## Write-Hand Man

*"Almost a Sidekick for CP/M"*

Ted Silveira—Computer Currents, Aug. 27, 1985

**"WHM is ingenious and works as intended"**

Jerry Pournelle, BYTE Magazine, Sept. 1985 (c) McGraw-Hill

**Now available for CP/M 2.2, CP/M 3.0 and ZRDOS!**

The convenience of *Sidekick* on your CP/M machine! Trigger **Write-Hand-Man** with a single keystroke and a window pops open to run desk accessories. Exit **Write-Hand-Man** and both the screen and program are restored. Use with any CP/M program and most any CP/M machine. Takes only 5K of memory.

### FEATURES

Notepad for quick notes  
Appointment calendar  
HEX calculator

File and Directory viewer  
Quick access phonebook  
14 digit decimal calculator

### BONUS

Add applications written by you or others! No other *Sidekick* lets you add applications. Dump screens, setup printers, communicate with other computers, display the date and time. Let your imagination run wild!

**\$49.95** (California residents add tax), shipping included. COD add \$2. Sorry, no credit cards or purchase orders. 30 day guarantee. Formats: 8 inch IBM, Northstar and most 5 inch (please specify).

**Write-Hand-Man** only works with CP/M 2.2, ZRDOS and CP/M 3.0 (please specify). Simple terminal configuration required. Not available for TurboDOS. Compatible with keyboard extenders, hard disks, and other accessories.

### Poor Person Software

3721 Starr King Circle  
Palo Alto, CA 94306  
415-493-3735

Trademarks: **Write-Hand-Man** — Poor Person Software, CP/M—Digital Research, **Sidekick**—Borland International

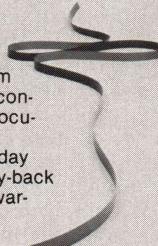
Circle no. 169 on reader service card.

# nine track tape users Micro to Mainframe Connection

The Model TC-50 1/2-inch tape subsystem provides a standard medium for transmission of mainframe data base information to PC users, while maintaining mainframe isolation and data integrity. Use ODI subsystems to import data to data base programs like dBase III.

The TC-50 subsystem also provides fast back-up capability as well as a device driver and interface software for popular compilers.

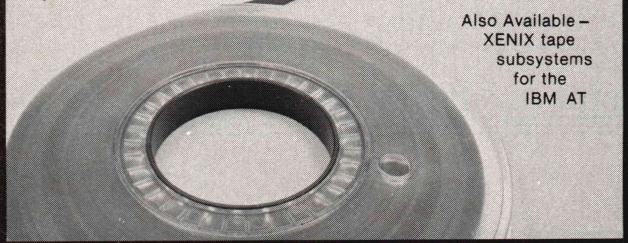
The TC-50 subsystem includes tape drive controller, cables and documentation. All ODI products carry a 30 day unconditional money-back guarantee, and are warranted for one year, parts and labor.



Overland Data, Inc.

5644 Kearny Mesa Road  
San Diego, CA 92111  
Tel. (619) 571-5555  
Telex 754923 OVERLAND

Also Available —  
XENIX tape  
subsystems  
for the  
IBM AT



Circle no. 192 on reader service card.

## RUN CP/M 80 ON IBM FAST!

Now your IBM PC or compatible can access the vast library of CP/M/80-2.2 software titles, and run them at lightning speed! Our revolutionary MICRUN/CPM non-emulating interface executes CP/M software on PC's by utilizing the power of NEC's newest microprocessor, the V-20. The V-20/30 chip replaces your PC resident 8088 or 8086 microprocessor chip, thereby providing 8 bit processing ability and actually improving its 16 bit performance by up to 50%. If your PC is not already equipped with the NEC V-20 there is no problem. Simply purchase our MICRUN/CPM software package, and we will provide you with the V-20 chip for only \$45 dollars.

### Features

- Incredibly fast-up to 8 MHz
- Ability to run CPM programs in color
- Cross sub-directory operations
- No expensive co-processor board req.
- Not an emulator
- Includes disk transfer utility for reading, writing, and formatting up to 70 different CP/M and MS-DOS disks.
- 16 logical/physical drives
- Includes RS232 communications program for transferring software to or from CPM system to PC.

### Supports

Osborne\* Kaypro\* Zenith H-19\* Televideo\* Radio Shack\* ADM3A\* VT-52\* Visual 210 adds Viewpoint\* Hazelton Espirit & 1510\* Plus many more

## ONLY \$99.95

Included in the above price: Interface software, disk transfer utility program, RS232 communications program, complete documentation. Not included is the NEC V-20 microprocessor chip which is available at a price of only \$45 dollars.



## 1-800-637-7226



Orders only.

Micro Interfaces Corporation  
6824 N.W. 169th Street  
Hialeah, Florida 33015  
(305) 823-8088



Dealer Inquiries Invited

Trademarks: CP/M (Digital Research, Inc.), IBM (IBM Corp.).

Circle no. 110 on reader service card.

# CRC

## LISTING TWO

(Listings continued, text begins on page 26)

```

{ start of validation testing }

PROCEDURE fultest( beg: INTEGER );
CONST
  maxpass = 5;
TYPE
  st40 = STRING[40];
  passtype = ARRAY[ 1..maxpass ] of INTEGER;
VAR
  adi, pass: INTEGER;
  adr, adt: passtype;

PROCEDURE overall( VAR a: passtype; t: INTEGER );
  VAR
    x, en, loc: INTEGER;
    data: BYTE;
  BEGIN
    FOR pass := 1 to maxpass DO
      BEGIN
        x := adi;
        CASE pass OF
          1: en := beg;
          2: en := beg + 1;
          3: en := beg + 127;
          4: en := beg + 511;
          5: en := beg + 81;
        END;
        FOR loc := beg TO en DO
          BEGIN
            { a source of data "random" enough for our purposes }
            { is available in the program code area }
            data := MEM[ Cseg: loc ];
            CASE t OF
              1: crcitby( x, data );
              2: crcfb3( x, data );
              3: crcitta( x, data );
              4: crctablu( x, data );
              5: mrcitt1( x, data );
              6: mrcitt3( x, data );
            END; { case }
          END;
          a[pass] := x;
        END;
      END; { overall }

PROCEDURE pbin( value: INTEGER );
  VAR
    i: INTEGER;
  BEGIN
    FOR i := 15 DOWNTO 0 DO
      BEGIN
        IF value < 0 THEN
          WRITE( '1' )
        ELSE
          WRITE( '0' );
        value := value SHL 1;
      END;
    END; { pbin }

PROCEDURE showres( s: st40 );
  VAR
    errfound: BOOLEAN;
  BEGIN
    errfound := FALSE;
    WRITE( '^M^J', s );
    FOR pass := 1 TO maxpass DO
      IF (adt[pass] <> adr[pass]) THEN
        BEGIN
          errfound := TRUE;
          WRITE( '^M^J' 'pass ', pass, ': CRC error.' );
          WRITE( '^M^J', 'Reference: ', pbin( adr[pass] ) );
          WRITE( '^M^J', 'Under Test: ', pbin( adt[pass] ) );
        END;
    IF NOT errfound THEN
      WRITE( ': No error.' );
    END; { showres }

BEGIN { fultest }
  WRITE( '^M^J' 'BEGIN Validation Testing.' );
  adi := -1;
  initctab;
  overall( adr, 1 );
  WRITE( '^M^J' 'Reference - crcitby (Pascal Bit-By-Bit).' );
  overall( adt, 2 );
  showres( 'crcfb3 (Pascal Fast Bit-By-Bit)' );
  overall( adt, 3 );
  showres( 'crcitta (Pascal Byte)' );
  overall( adt, 4 );
  showres( 'crctablu (Pascal Table)' );
  overall( adt, 5 );
  showres( 'mrcitt1 (Machine Code Byte)' );
  overall( adt, 6 );
  showres( 'mrcitt3 (Machine Code Table)' );
  WRITELN( '^M^J' 'END Validation Testing.' );
  END; { fultest }

```

```

{ start of timing }

PROCEDURE timetest( loops: INTEGER );
{ organize the time testing of various routines }
VAR
  a, b: timearty;
  i, x: INTEGER;
  by: BYTE;
  empty: REAL; { time for empty loops }
  calltime: REAL; { time for loops and empty procedure }

PROCEDURE crcmt( VAR dx: INTEGER; data: INTEGER );
BEGIN
END;

PROCEDURE showtimedif( a, b: timearty; c: INTEGER );
{ display a line of time results }
VAR
  dif, Noloop, NoloopNocall: REAL;
BEGIN
  dif := timedif( a, b );
  Noloop := dif - empty;
  NoloopNocall := dif - calltime;
  WRITELN(
    Noloop:7:3, ' ', NoloopNocall:7:3, ' ', Noloop * (1000.0 / c):7:3 );
END; { showtimedif }

BEGIN { timetest }
  calltime := 0.0; { the global }
  by := $A5; { your typical data byte }

  WRITELN(
    '^M^J^J' Turbo Pascal runs CRC-CCITT on 8088 under Bare MSDOS',
    '^M^J' FOR ', loops,' OPERATIONS: 7.16 MHz CLOCK
    (multiply by 1.5 for 4.77 MHz)');

  WRITE( '^M^J'Empty loop: ' );
  readhms( a );
  FOR i := 1 TO loops DO
    ;
  readhms( b );
  empty := timedif( a, b ); { the global }
  WRITE( empty:5:3, ' secs' );

  WRITE( '^M^J'Empty procedure in loop: ' );
  readhms( a );
  FOR i := 1 TO loops DO
    crcmt( x, by );
  readhms( b );
  calltime := timedif( a, b ); { another global }
  WRITE( calltime:5:3, ' secs' );

  WRITE( '^M^J' (procedure overhead alone -',
    (calltime - empty) * (1000.0 / loops):6:3,
    ' msec each)' );

  WRITELN(
    '^M^J^J' ', loops:5, ' Uses (secs) 1 Use
    Procedure In Line Procedure In Line' );

  WRITE(
    '^M^J' Pascal Bit-by-Bit: ' );
  readhms( a );
  FOR i := 1 TO loops DO
    crcitby( x, by );
  readhms( b );
  showtimedif( a, b, loops );

  WRITE(
    '^M^J' Pascal Fast B-B-B: ' );
  readhms( a );
  FOR i := 1 TO loops DO
    crccbbb( x, by );
  readhms( b );
  showtimedif( a, b, loops );

  WRITE(
    '^M^J' Pascal Byte: ' );
  readhms( a );
  FOR i := 1 TO loops DO
    crccitta( x, by );
  readhms( b );
  showtimedif( a, b, loops );

  WRITE(
    '^M^J' Pascal Table: ' );
  readhms( a );
  FOR i := 1 TO loops DO
    crctablu( x, by );
  readhms( b );
  showtimedif( a, b, loops );

  WRITE(
    '^M^J' Machine Code Byte: ' );
  readhms( a );
  FOR i := 1 TO loops DO
    mrccttbl( x, by );
  readhms( b );
  showtimedif( a, b, loops );

  WRITE(
    '^M^J' Machine Code Table: ' );
  readhms( a );
  FOR i := 1 TO loops DO
    mrccttbl3( x, by );
  readhms( b );
  showtimedif( a, b, loops );

```

```

WRITELN;
END; { timetest }

PROCEDURE datatimes;
{ data character times, for comparison }
CONST
  rates: ARRAY[1..7] of INTEGER =
    ( 12, 24, 48, 96, 192, 384, 576 );
VAR i: INTEGER;
BEGIN
  WRITELN( '^M^J^J'Character Times for Various Bit Rates: ' );
  WRITE( '^M^J' bits/sec char/sec msec/char );
  FOR i := 1 TO 7 DO
    WRITE( '^M^J, '' :4, (100.0 * rates[i]):5:0, '' :8,
    (10.0 * rates[i]):5:0, '' :8,
    (100.0 / rates[i]):5:3 );

  WRITELN;
END; { datatimes }

BEGIN { main: crctime }
WRITELN;

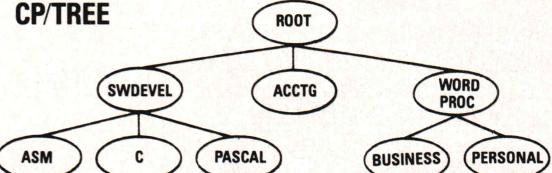
ConOutPtr := OFS( bdosch ); { output through MSDOS }

ownership;
fultest( 100 );
timetest( 10000 );
datatimes;
END.

```

## End Listings

### CP/TREE



### Tree-Structured Named Directories for CP/M 2.2

- Transforms user areas into Unix-like directories
- Provides Unix-like directory commands: CD, MKDIR, PWD, RMDIR, TREE
- Includes a CCP replacement featuring:
  - Command and file search path. Use programs like WordStar from any directory!
  - Erase with query
  - Wildcard rename with query
- Provides output redirection to disk file
- Uses as little as  $1/2$  k RAM, never more than  $2\frac{1}{4}$  k
- A must for hard disks
- Installs easily: requires no modifications to BDOS or BIOS
- Requires standard CP/M 2.2 (not 3.0 or Apple), Z80, 48k RAM

**\$29.95 plus \$4.00 s&h**

**To order:** Specify disk format (8" SSSD, NorthStar DD. Call for info on others.). MC, Visa, COD (add \$1.90), check (delays shipping 2 weeks). MA residents add 5% sales tax. POs not accepted.

**Precise Electronics**  
 P.O. Box 339  
 New Town Branch  
 Boston, MA 02258  
 tel: (617) 332-3977

Apple<sup>®</sup> Apple Computer. CP/M<sup>®</sup> Digital Research. WordStar<sup>®</sup>  
 MicroPro International. Z80<sup>®</sup> Zilog. Unix<sup>®</sup> AT&T Technologies.

# HISOFT

HIGH QUALITY  
SOFTWARE

CP/M

HiSoft has been selling Z80 CP/M software in Britain and Europe for over 4 years. Now we'd like to introduce **you** to our range of programming languages:

**HiSoft Devpac:** Z80 assembler/editor/ debugger

**HiSoft C:** Kernighan/Ritchie implementation

**HiSoft Pascal:** fast, standard compiler  
All at \$69 inclusive each.

These programs are also available for other Z80 machines including Timex 2068.

Call or write for full technical details and press commentaries, or order from:

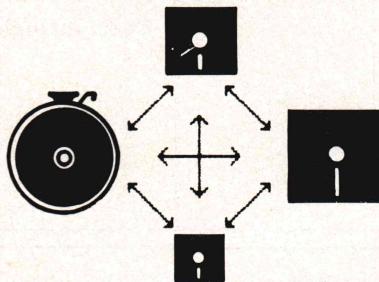


**HiSoft**

180 High St. North  
Dunstable LU6 1AT  
ENGLAND  
01144 (582) 696421

Circle no. 134 on reader service card.

## DATA CONVERSION



TRANSFER DATA BETWEEN OVER  
500 DIFFERENT COMPUTER SYSTEMS

WORD PROCESSORS TOO

QUICK TURN-AROUND

PRICES FROM \$9 PER DISK

CALL OR WRITE FOR YOUR

**FREE CATALOG**

**PORT-A-SOFT**

555 S. STATE ST., SUITE #12  
P.O. BOX 1685, OREM, UT 84057  
(801) 226-6704

Circle no. 229 on reader service card.

## Dr. Dobb's Journal

Subscription  
Problems?

No Problem!



Give us a call and we'll  
straighten it out. Today.

Outside California

CALL TOLL FREE: 800-321-3333

Inside California

CALL: 619-485-6535 or 6536

## PASCAL POWERS

### LISTING ONE (Text begins on page 36)

```

{POWERN.PLB #1.08 85-11-18  INTEGER POWERS OF REALS
V01 L08 85-11-18 presentation streamlining by DEH.
L03 update on 85-06-18 by DEH to eliminate one
always-trivial multiplication by expanding
on an idea of David Gries.
L00 created on 85-06-05 by Dennis E. Hamilton
to answer a number of electronic requests
for help with Pascal powers. }

function
  PowerN(x: real;  {initial value: x0}
         n: integer {required exponent} )
         :real {value of x0**n};

  var i: integer {non-negative power of x remaining
                 to be computed};
      r: real {running intermediate-result value};

  BEGIN {PowerN}
    if n = 0
    then PowerN := x/x
    else begin (n < 0)
      i := abs(n);
      {x**i = x0**abs(n)}
      while not odd(i)
        do begin {((x*x)**(i div 2)) = x**i}
          x := sq(x); i := i shr 1;
        end;
      r := x;
      {maintain odd(i) and r*x**(i-1) = x0**abs(n),
      taking r*x**(i-1) = r upon reaching i = 1}
      while i < 1
        do begin
          repeat
            {((x*x)**(i div 2)) = x**(i - i mod 2)}
            x := sgr(x); i := i shr 1;
            until odd(i);
          {odd(i) and r*x**i = x0**abs(n)}
          r := r*x;
        end;
      {finally i = 1 and r = x0**abs(n)}
      if n < 0
      then PowerN := 1.0/r
      else PowerN := r;
    end;
  END {PowerN = x0**n};

```

End Listing One

### LISTING TWO

```

program TPWRN(out);
{TPWRN.PAS #1.08 85-11-18 TEST POWERN.PLB CALCULATION OF POWERS
V01 L08 pretty-print update on 85-09-15 with tidier test output.
L02 update on 85-06-20 by DEH to exhibit precision-maintenance
difficulties with (1/7)**-i and exp(i*ln(7)) variations.
L00 created on 85-06-05 by Dennis E. Hamilton, just for
simple confirmation of POWERN's method.}

var   i: integer {counter of trial exponents};
      p7: real {intermediate power of 7 to be checked};
      ln7: real {logarithm of 7 used in comparison with exp(ln) method};
      rv7: real {value of 1/7 used in showing precision loss};
      out: text {file variable used for direction of output as needed};

{SI POWERN.PLB } {Vintage 1.00}
{Include PowerN here by whatever method the Pascal system supports.
The above SI pragmat, used with Borland International's Turbo Pascal,
causes a copy of POWERN.PLB to be included at this point.}

BEGIN {Testing the basic features of POWERN.PLB}
assign(out, 'CON:');
{Change to a disk-file name when you want to capture the report in
a file for comparison, uploading, etc. Implementation-dependent.}
rewrite(out);
writeln(out, 'TPWRN> #1.08 85-11-18 TEST OF POWERN FUNCTION RESULTS');
writeln(out, '.....1.....2.....3.....4.....5.....');
writeln(out, '.....i.....(-1)**i.....p = 7**i.....(1/7)**-i - p');
writeln(out, '.....exp(i*ln(7)) - p7');

ln7 := ln(7.0); rv7 := 1.0/7.0;
for i := 0 to 17
do begin
  write(out, 1:8, PowerN(-1,i) :11:0);
  p7 := PowerN(7.0,i);
  {It is important to use a number that is prime to the floating-point
  radix. Although 5 is easier to check mentally, it doesn't show enough
  about accuracy when decimal arithmetic is used, as with Turbo-BCD.}
  write(out, p7:16:0);
  write(out, PowerN(rv7,-i) - p7:20:13);
  {Use the known-to-be-inexact reciprocal to show how errors magnify}
  writeln(out, exp(i*ln7) - p7:21:13);
  {Show deviation of the inefficient exp(i*ln(7)) on the same basis}
end;
close(out);
END.

```

TPWRN&gt; MODIFIED #1.02 85-06-20 TEST OF POWERN FUNCTION RESULTS

I	(-1)**I	p = 7**I	(1/7)**-I - p	exp(I*ln(7)) - p
0	1	1	0.0	0.0
1	-1	7	0.0	-0.0000000000 146
2	1	49	0.0000000000 582	-0.0000000000 3492
3	-1	343	0.0000000000 9313	-0.0000000000 32596
4	1	2401	0.0000000000 74506	-0.0000000000 298023
5	-1	16807	0.0000000000 596046	-0.0000000000 3874302
6	1	117649	0.0000000000 8344650	-0.0000000000 26226044
7	-1	823543	0.0000000000 57220459	-0.0000000000 362396240
8	1	5764801	0.0000000000 457763672	-0.0000000000 1296997070
9	-1	40353607	0.0000000000 3662109375	-0.0000000000 9155273438
10	1	282475249	0.0000000000 29296875000	-0.0000000000 126953125000
11	-1	1977326743	0.0000000000 234375000000	-0.0000000000 859375000000
12	1	13841287201	0.0000000000 187500000000	-0.0000000000 625000000000
13	-1	96889010407	0.0000000000 1.37500000000	-0.0000000000 250000000000
14	1	678223072850	0.0000000000 11.0000000000	-0.0000000000 58.0000000000
15	-1	4747561509900	0.0000000000 80.0000000000	-0.0000000000 400.0000000000
16	1	33232930570000	0.0000000000 640.0000000000	-0.0000000000 1472.0000000000
17	-1	232630513990000	0.0000000000 4864.0000000000	-0.0000000000 20224.0000000000

These results were obtained by assign(out, 'TPWRN.PRN') and compiling with POWERN.PLB #1.03. CP/M-80 Turbo Pascal version 3.00A was used. The report file was then incorporated into this edition of TPWRN.PAS as part of final editing as a MicroPro WordStar document. (The WordStar edition is converted back to a verified compilable form by "printing" to disk and filtering out all printing-control information using a simple utility program.) The numbers in the error columns have been doctored by addition of spacing to make the growth of differences easier to discern.

Similar results should be obtained using MS-DOS, CP/M-86 and IBM PC versions of Turbo Pascal. Turbo-8087 should obtain better results in all columns because of the greater precision maintained internally. Turbo-BCD should also show improvements, with decreased speed, after the last column is either dropped or library procedures for BCD ln() and exp() are obtained. (For Turbo-BCD and Turbo-8087 both, it is instructive to increase the number of decimal positions in the last two columns in order to see what error there is, however much smaller it turns out to be.)

(\* end of TPWRN.PAS \*)

End Listings

# I Q C L I S P

## THE CLOSEST THING TO COMMON LISP AVAILABLE FOR YOUR PC

### RICH SET OF DATA TYPES

Bignums, for high precision arithmetic  
8087 support, for fast floating point  
Arrays, for multidimensional data  
Streams, for device-independent i/o  
Packages, for partitioning large systems  
Characters, strings, bit-arrays

### FULL SET OF CONTROL PRIMITIVES

flet, labels, macrolet, for local functions  
if, when, unless, case, cond, for conditionals  
Keyword parameters, for flexibility  
Multiple-valued functions, for clarity  
Flavors, for object-oriented programming  
Stacks, for coroutining  
Closures, for encapsulation

### LARGE COMPLEMENT OF FUNCTIONS

Mappers, for functional programming  
format, for output control  
sort, for user-specified predicates  
Transcendental floating point functions  
String handling functions  
Over 400 functions altogether

### APPLICATION SUPPORT

Save and restore full environments  
User-specified initializations  
Assembly language interface

### HARDWARE REQUIREMENTS

8088 or 8086 CPU, MSDOS Operating System  
390K RAM or more

### IQCLISP

5 1/4" diskettes  
and manual

\$300.00

Foreign orders add \$30.00 for airmail.  
U.S. Shipping included for prepaid orders.

*fq Integral Quality*

P.O. Box 31970  
Seattle, Washington 98103  
(206) 527-2918

Washington State residents add sales tax.  
VISA and MASTERCARD accepted.

### EXTENDABILITY

defstruct, to add data types  
Macros, to add control constructs  
Read macros, to extend the input syntax  
Extendable arithmetic system  
Customizable window system

### DEBUGGING SUPPORT

step, for single-stepping  
trace, for monitoring  
break, for probing  
inspect, for exploring  
Flexible error recovery  
Customizable pretty-printer

### MSDOS INTERFACE

Random access files  
Hierarchical directory access  
MSDOS calls

### DOCUMENTATION

On-line documentation of functions  
apropos  
300-page indexed reference manual

# We will do whatever it takes to make DSD86 the best debugger available for the IBM PC.

For starters, we have by far the best design, a superior base to build from.

While the competition adds new "modes" for every feature, we have a pure, consistent and expandable design. While the competition forces you to accept their particular philosophy, we offer maximized flexibility. If you already have a debugger or are looking for your first, look no further because you can't do any better. We invite you to compare our debugger, DSD86, with any other on the market.

- Recursive Command Macros & Files ■
- Bind Macros to any key ■
- Multi-segment Symbol Support ■
- Symbolic Register & Stack Displays ■
- User Customizable Screen Layout ■
- Superior Mode-less Design ■
- Source Window for MS Languages ■
- User Writable Commands & Displays ■
- Fast Screen Update ■
- Unique Breakpointing Facilities ■
- 30 Day Money Back Guarantee ■

Call or write for our free report on truly advanced debugging technology which explains DSD86's design and why it is superior to the debugger you are currently using.

Take the DSD challenge: secure a money back guarantee with any of our competitors. Buy both debuggers and use them for a month. Send the one you like least back for a refund.

## Only \$69.95!

**Soft Advances**  
P.O. Box 49473  
Austin, Texas 78765  
**512-478-4763**

"Programming for Productivity and Profit"  
Please include \$4 shipping

Circle no. 83 on reader service card.

# ADA

## LISTING ONE (Text begins on page 42)

LISTING 1 - Draw Poker Program written in Ada

```

begin
  Open_New(STOCK);
  loop
    put("How many dollars do you want to bet? "); get(WAGER);
    exit when WAGER = 0;
    Shuffle(STOCK);
    Open_New(PLAYERS_HAND);
    for I in 1 .. 5 loop
      Deal_A_Card(PLAYERS_HAND, STOCK);
    end loop;
    put(PLAYERS_HAND);
    Discard_From(PLAYERS_HAND);
    loop
      exit when Filled(PLAYERS_HAND);
      Deal_A_Card(PLAYERS_HAND, STOCK);
    end loop;
    put(PLAYERS_HAND);
    VALUE := Value_Of(PLAYERS_HAND);
    case VALUE is
      when ROYAL_FLUSH => PAYOFF := 250;
      when STRAIGHT_FLUSH => PAYOFF := 50;
      when FOUR_OF_A_KIND => PAYOFF := 25;
      when FULL_HOUSE => PAYOFF := 6;
      when FLUSH => PAYOFF := 5;
      when STRAIGHT => PAYOFF := 4;
      when THREE_OF_A_KIND => PAYOFF := 3;
      when TWO_PAIR => PAYOFF := 2;
      when others => PAYOFF := 0;
    end case;
    if PAYOFF = 0
      then put_line("Sorry, you lose.");
      else put("You have "); put(VALUE); put("!");
      put("You win"); put(WAGER*PAYOFF); put_line(" dollars!");
    end if;
  end loop;
end Draw_Poker;

```

**End Listing One**

## LISTING TWO

LISTING 2 - The general form of a procedure

```

procedure *1 is
  *2
  begin
    *3
  exception
    *4
  end *5;

```

**End Listing Two**

## LISTING THREE

LISTING 3 - The Open\_New procedure

```

procedure Open_New(DECK : out Decks) is
  i : integer := 0;
  CARD : Cards;
begin
  for S in Suits loop
    for R in Ranks loop
      CARD.SUIT := S;
      CARD.RANK := R;
      i := i + 1;
      DECK.FAN(i) := CARD;
    end loop;
  end loop;
  DECK.CARDS_LEFT := i;
  if i /= CARDS_IN_DECK then raise DECK_ERROR; end if;
exception
  -- CONSTRAINT_ERROR or DECK_ERROR may be raised by this
  -- procedure If the number of cards in a deck does not
  -- equal the number of cards generated.
  when DECK_ERROR | CONSTRAINT_ERROR =>
    raise DECK_ERROR; -- convert all errors to DECK_ERROR;
end Open_New;

```

**End Listing Three**

## LISTING FOUR A

Listing 4 (Part A) - PLAYING\_CARDS package specification

```

--                                                 CARDS.ADA
--                                                 19 JULY 1984
--                                                 DO-WHILE JONES
package PLAYING_CARDS is
  CARDS_IN_DECK : constant integer := 52;
  CARDS_IN_HAND : constant integer := 5;
  DECK_ERROR    : exception; -- raised by Open_New
  DECK_EXHAUSTED : exception; -- raised by Deal_A_Card
  HAND_FULL     : exception; -- raised by Deal_A_Card

```

```

type Suits is (CLUBS, DIAMONDS, HEARTS, SPADES);
type Ranks is (TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN,
  JACK, QUEEN, KING, ACE);

type Cards is
  record
    SUIT : Suits;
    RANK : Ranks;
  end record;

type Fans is array(integer range <>) of Cards;
type Status is array(integer range <>) of boolean;
type Decks is
  record
    CARDS_LEFT : integer;
    FAN : Fans(1..CARDS_IN_DECK);
  end record;

type Hands is
  record
    PLAYED : Status(1..CARDS_IN_HAND);
    FAN : Fans(1..CARDS_IN_HAND);
  end record;

function Card_Number(X : integer; HAND : Hands) return Cards;
function Played_Card_Number(X : integer; HAND : Hands) return boolean;
function Suit_of(CARD : Cards) return Suits;
function Rank_of(CARD : Cards) return Ranks;

procedure put(SUIT : Suits);
procedure put(RANK : Ranks);
procedure put(CARD : Cards);
procedure put(HAND : Hands);

procedure Open_New(DECK : out Decks);           -- create a new deck
procedure Shuffle(DECK : in out Decks);         -- shuffle a deck

procedure Open_New(HAND : out Hands);           -- create a new hand
procedure Sort(HAND : in out Hands);           -- sort by rank, ignore suits
procedure Discard_From(HAND : in out Hands);
function Filled(HAND : Hands) return boolean; -- is the hand full?

procedure Deal_A_Card(HAND : in out Hands; DECK : in out Decks);
end PLAYING_CARDS;

```

**End Listing Four A**

*(Continued on next page)*

# WINDOWS FOR DATA™

## Featuring One-Step Data Entry\*

Now you can code fast, powerful data entry windows, improve user convenience - reduce input errors.

All the power, convenience and flexibility of the #1 window utility for the IBM PC. Our WINDOWS FOR C™ combined with a professional window-based data entry system.

Complete control over screen display and entry of data within a convenient flexible window environment.

WINDOWS FOR C    WINDOWS FOR DATA  
(Includes WINDOWS FOR C)

<b>PCDOS</b>	<b>\$ 195</b>	<b>\$ 295</b>
<b>PC/XENIX</b>	<b>\$ 395</b>	<b>\$ 595</b>
<b>UNIX</b>	<b>CALL</b>	<b>CALL</b>

Full source available. Master Card & Visa accepted. Shipping \$ 3.50. VT residents add 4% tax.

WINDOWS FOR DATA™ provides versatile, easy-to-use data entry functions that operate within windows.

### CAPABILITIES INCLUDE:

- Pop-up data entry windows
- Multiple field types
- Data validation functions
- Field-specific & context-sensitive help
- Lotus-style menu design
- Single field entry option
- Date, time and string utilities
- Dynamic control of data-entry environment
- ◆ User input to data-structure variables without intervening code.



**Vermont  
Creative  
Software**

21 Elm Ave.  
Richford, VT 05476  
802-848-7738, ext. 31

Circle no. 157 on reader service card.



# The C Users' Group

Over 75 volumes of public domain software including:

- compilers
- editors
- text formatters
- communications packages
- many UNIX-like tools

Write or call for more details

## The C Users' Group

Post Office Box 97  
McPherson, KS 67460  
(316) 241-1065

Circle no. 181 on reader service card.

**NEW!**

## Advanced Trace86™

Symbolic Debugger & Assembler Combo

Full-screen trace with single stepping;  
Even backstepping!  
Write & Edit COM & EXE programs  
Conditional breakpoints (programmable)  
Switch between trace and output screen;  
Or set up two monitors  
8087, 80186, 80286, 80287 support  
Write labels & comments on code  
Polish hex/decimal calculator  
and more ... Priced at \$175.00

To order or request more information contact:

**M** Morgan Computing Co., Inc.  
2520 Tarpley Rd. Suite 500  
(214) 245-4763 Carrollton, TX 75006

Circle no. 128 on reader service card.

## UNIVERSAL CROSS-REFERENCER

- **IT WORKS WITH ALL LANGUAGES**  
BASIC, C, PASCAL, FORTRAN, COBOL, ASSEMBLER, dBASE ... you name it.
- **IT HANDLES** standard languages, extended languages & exotic languages.
- **IT CONFIGURES** to your compiler, interpreter, or assembler — ALL BRANDS — ALL VERSIONS.

**INTRODUCTORY PRICE \$39.95**

\$3.00 S/H - MC/Visa/Check - 6% Texas Sales Tax  
For the IBM PC, XT & compatibles. DOS 2.0+

**DALSOFT SYSTEMS**  
3565 High Vista - Dallas, TX 75234  
(214) 247-7695

Circle no. 86 on reader service card.

# ADA

## LISTING FOUR B (Listing continued, text begins on page 42)

LISTING 4 (Part B) - PLAYING\_CARDS package body

```

-- CARD.B.ADA
-- 19 JULY 1984
-- DO WHILE JONES

with CON_IO; use CON_IO;
with APL; use APL;
package body PLAYING_CARDS is

CONSTRAINT_ERROR : exception; -- required only by Maranatha A

function Card_Number(X : integer; HAND : Hands) return Cards is
begin
  return HAND.FAN(X);
end Card_Number;

function Played_Card_Number(X : integer; HAND : Hands) return boolean is
begin
  return HAND.PLAYED(X);
end Played_Card_Number;

function Suit_of(CARD : Cards) return Suits is
begin
  return CARD.SUIT;
end Suit_of;

function Rank_of(CARD : Cards) return Ranks is
begin
  return CARD.RANK;
end Rank_of;

procedure put(SUIT : Suits) is
begin
  case SUIT is
    when CLUBS => put("CLUBS");
    when DIAMONDS => put("DIAMONDS");
    when HEARTS => put("HEARTS");
    when SPADES => put("SPADES");
  end case;
end put;

procedure put(RANK : Ranks) is
begin
  case RANK is
    when TWO => put("TWO");
    when THREE => put("THREE");
    when FOUR => put("FOUR");
    when FIVE => put("FIVE");
    when SIX => put("SIX");
    when SEVEN => put("SEVEN");
    when EIGHT => put("EIGHT");
    when NINE => put("NINE");
    when TEN => put("TEN");
    when JACK => put("JACK");
    when QUEEN => put("QUEEN");
    when KING => put("KING");
    when ACE => put("ACE");
  end case;
end put;

procedure put(CARD : Cards) is
  RANK : Ranks;
  SUIT : Suits;

begin
  put(Rank_of(CARD)); put(" of "); put(Suit_of(CARD));
end put;

procedure put(HAND : Hands) is
begin
  for i in 1..CARDS_IN_HAND loop
    if Played_Card_Number(i, HAND) then
      null; -- don't display a card that isn't there
    else
      put(Card_Number(i, HAND));
      put("  "); -- separate cards with two blanks
    end if;
  end loop;
  new line;
  end_put;
end;

procedure Open_New(DECK : out Decks) is
  i : integer := 0;
  CARD : Cards;
begin
  for S in Suits loop
    for R in Ranks loop
      CARD.SUIT := S;
      CARD.RANK := R;
      i := i + 1;
      DECK.FAN(i) := CARD;
    end loop;
  end loop;
  DECK.CARDS_LEFT := i;
  if i /= CARDS_IN_DECK then raise DECK_ERROR; end if;
exception
  -- CONSTRAINT_ERROR or DECK_ERROR may be raised by this
  -- procedure If the number of cards in a deck does not
  -- equal the number of cards generated.
  when DECK_ERROR | CONSTRAINT_ERROR =>

```

```

raise DECK_ERROR; -- convert all errors to DECK_ERROR;
end Open_New;

procedure Shuffle(DECK : in out Decks) is
  SEQUENCE : Random_Sequence(1..CARDS_IN_DECK);
  TEMP : DECKS;
begin
  TEMP.CARDS_LEFT := CARDS_IN_DECK;
  SEQUENCE := Deal(CARDS_IN_DECK, CARDS_IN_DECK);
  for i in 1..CARDS_IN_DECK loop
    TEMP.FAN(i) := DECK.FAN(SEQUENCE(i));
  end loop;
  DECK := TEMP;
end Shuffle;

procedure Deal_A_Card(HAND : in out Hands; DECK : in out Decks) is
  X : integer := 0;
begin
  -- find an empty slot in the hand
  loop
    X := X+1;
    if X > CARDS_IN_HAND then raise HAND_FULL; end if;
    exit when Played_Card_Number(X, HAND);
  end loop;
  -- draw a card from the deck and put it in the empty slot
  if DECK.CARDS_LEFT < 1
    then raise DECK_EXHAUSTED;
    else DECK.CARDS_LEFT := DECK.CARDS_LEFT-1;
  end if;
  HAND.FAN(X) := DECK.FAN(CARDS_IN_DECK - DECK.CARDS_LEFT);
  HAND.PLAYED(X) := FALSE;
end Deal_A_Card;

procedure Sort(HAND : in out Hands) is
  SORTED : boolean;
  TEMP : Cards;
begin
  loop
    SORTED := TRUE;
    for i in 1..CARDS_IN_HAND loop
      if Rank_of(Card_Number(i, HAND)) > Rank_of(Card_Number(i+1, HAND)) then
        TEMP := Card_Number(i, HAND);
        HAND.FAN(i) := Card_Number(i+1, HAND);
        HAND.FAN(i+1) := TEMP;
        SORTED := FALSE;
      end if;
    end loop;
    exit when SORTED;
  end loop;
end Sort;

```

(Continued on next page)

## SUPER-CHARGE your directories for only \$39.95

Say goodbye to path names and unleash the full power of DOS on your IBM Personal Computer. With SuperPATH™ from Martin Scot, you can run any program from any directory, from any disk, anytime.

### SEE THE BIG PICTURE.

Tired of playing hide and seek with your files? SuperPATH tells you where your files are, instantly. Anywhere on your disk.

### EXPLOSIVE PERFORMANCE.

Program your directories. To display just the files you want to see, in any format, sorted or double sorted. To set up your keyboard macros for each new directory you enter. To back up your files automatically when you leave a directory.

Or, you can use SuperPATH's batch building option to create your own file management utilities.

### NO RISK.

Comes with a 30 day money-back guarantee.

**CALL (206) 527-9605 to order SuperPATH.**

**MARTIN SCOT**

4515 Purdue N.E. Seattle, WA 98105

Circle no. 167 on reader service card.

## GRAF 3.0

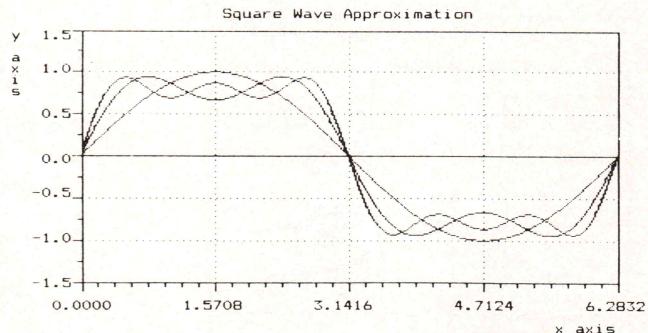
the complete  
BUSINESS and SCIENTIFIC  
printer graphics program

- display floating point data directly from spreadsheets, data bases, word processors, and programming languages (or the keyboard) in a wide variety of bar, pie, line, and scatter plots
- menu driven operation supports automatic graph scaling, labeling, and legend creation

**\$49.95** **\$69.95**

CP/M-80 MS-DOS / PC-DOS

- plot and group up to 6 different variables on a single graph, distinguished by up to 14 different fill-in patterns and 8 different point-plotting symbols
- add up to 5 different-density grid lines, and choose from a wide variety of numerical labeling options



**GRAF 2.0 Update Policy:** Returning your original GRAF 2.0 disk to MSC entitles you to \$20.00 off the above price.

TERMS: WE ship via first class mail. The above prices include domestic shipping and handling. Orders outside USA require additional \$5.00 for postage. N.Y. residents add state sales tax. When ordering you MUST state your computer and printer make and model. We support MS-DOS (PC-DOS) version 2.0 or later on computers with at least 192K RAM, and CP/M-80 version 2.2 or later on 280 computers (other than modified Apple) supporting a TPA of at least 54K (requires 64K of RAM). Most soft-sector disk formats are available. (If you can read several formats, please send us a list.) GRAF 3.0 works with any printer fully compatible with one of the following: Epson FX, RX, LX, MX (with GRAFTRAX), or LQ-1500, C, Itho Prowriter, NEC 803A, Star Micro, Gemini 10K, 15K, SG-10, SG-15, IBM Graphics Printer, Okidata 92, and earlier Okidata models equipped with the "IBM Plug 'n' Play" chips. (If you have an Okidata printer, other than the 92, the Plug 'n' Play chips are required.)

**MSC**

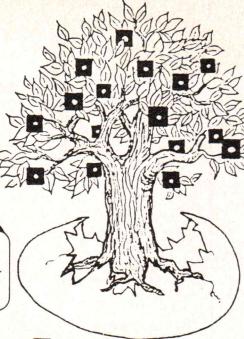
Microcomputer  
Systems  
Consultants

27 Forest Avenue Port Jefferson Station New York 11776-1820

Circle no. 136 on reader service card.

# Tree Shell

A Graphic  
Visual Shell for  
Unix/Xenix  
End-Users and  
Experts Alike!



## COGITATE

### "A Higher Form of Software"

24000 Telegraph Road  
Southfield, MI 48034  
(313) 352-2345  
TELEX: 386581 COGITATE USA

Circle no. 81 on reader service card.

## ICs PROMPT DELIVERY!!!

SAME DAY SHIPPING (USUALLY)

OUTSIDE OKLAHOMA: NO SALES TAX

V20	CPU μPD70108D-8	\$16.00
8087-2	Math Coprocessors	140.00
<b>DYNAMIC RAM</b>		
256K	64Kx4 150 ns	\$4.75
256K	256Kx1 120 ns	3.25
256K	256Kx1 150 ns	2.47
128K	128Kx1 150 ns	3.50
64K	16Kx4 150 ns	2.75
64K	64Kx1 150 ns	1.00
<b>EPROM</b>		
27C256	32Kx8 250 ns	\$7.50
27256	32Kx8 250 ns	4.75
27C64	8Kx8 200 ns	3.75
2764	8Kx8 250 ns	2.50
<b>STATIC RAM</b>		
6264LP-15	8Kx8 150 ns	\$2.99
OPEN 7 DAYS: WE CAN SHIP VIA FED-EX ON SAT.		
MasterCard/VISA or UPS CASH COD		
<b>Factory New, Prime Parts μP∞</b>		
MICROPROCESSORS UNLIMITED, INC.		
24,000 S. Peoria Ave., BEGGS, OK 74421 (918) 267-4961		
Prices shown above are for Dec. 2, 1985		
NO EXTRA COST FOR FED-EX DELIVERY ON ORDERS RECEIVED BY 12 NOON		
THURSTON AIR FR. P.O. ONE		
640 Kbyte MOTHERBOARD KITS: IBM PC/XT		
640 Kbyte 150 & 100, Compat. Portable & Plus: \$64.44		
QUANTITY ONE PRICES SHOWN		
NEW: 100 ns 256Kx1 D-RAM @ \$5.15		

MasterCard/VISA or UPS CASH COD  
**Factory New, Prime Parts μP∞**  
MICROPROCESSORS UNLIMITED, INC.  
24,000 S. Peoria Ave., BEGGS, OK 74421 (918) 267-4961

Prices shown above are for Dec. 2, 1985

Please call for current prices. Prices subject to change. Please expect higher or lower prices on some parts due to supply & demand and our changing costs. Shipping & insurance extra. Cash discount prices shown. Orders received by 6 PM CST can usually be delivered to you by the next morning, via Federal Express Standard Air (\$6.00, or Priority One (\$13.00).

Circle no. 105 on reader service card.

## TOTAL RECALL

FOR \$104.95

Now you can retrieve everything you entered during program development -- from version one thru all updates. SRMS does it with a complete source utility at an affordable price.

You can retrieve specific versions of a program, make changes and reinstate the source while recording when, why and where changes were made.

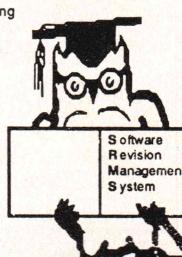
All versions are stored in a single library without duplication of common code or text -- saves disk space. You can also store comments specific to each version -- provides complete development history and documentation of the program.

SRMS supports recall and edit of programs written in BASIC, FORTRAN, PASCAL (including Turbo Pascal), C and ASSEMBLY CODE.

Requires MS or PC DOS, Version 2 and up with 128K and one disk (floppy or hard).

7048 Stratford Road  
Woodbury, MN 55125  
(612) 739-4650

Quilt Computing



Software  
Revision  
Management  
System

Circle no. 107 on reader service card.

# ADA

## LISTING FOUR B (Listing continued, text begins on page 42)

```

procedure Open_New(HAND : out Hands) is
begin
  for i in 1..CARDS IN HAND loop
    HAND.PLAYED(i) := TRUE; -- hand is empty (all cards have been played)
  end loop;
end Open_New;

procedure Discard_From(HAND : in out Hands) is
  RESPONSE : character;
begin
  for i in 1..CARDS IN HAND loop
    put("Do you want to discard the ");
    put(Card Number(i, HAND));
    put(" (Y/N) ");
    get(RESPONSE); new line;
    if RESPONSE = 'Y' or RESPONSE = 'y'
      then HAND.PLAYED(i) := TRUE;
    end if;
  end loop;
end Discard_From;

function Filled(HAND : Hands) return boolean is
begin
  for i in 1..CARDS IN HAND loop
    if Played Card Number(i, HAND) then
      return FALSE; -- if any card is played, hand is not filled
    end if;
  end loop;
  return TRUE; -- if no cards played, hand is filled
end Filled;

end PLAYING_CARDS;

```

End Listing Four B

## LISTING FIVE A

LISTING 5 (Part A) - APL package specification

```

-- APLS.ADA
-- 20 JULY 1984
-- DO WHILE JONES

-- This package simulates some APL functions.

-- Roll(X) returns a random integer in the range 1..X.

-- Deal(X,Y) returns a random sequence of X elements all
--   of which are in the range 1..Y. No element appears
--   twice in the random sequence.

```

```

package APL is

  subtype positive is integer range 1..integer'last;
  -- The above line is not required in Ada.
  -- (It is required for Maranatha A.)

  type Random_Sequence is array(positive range <>) of positive;
  function Roll(LIMIT : positive) return positive;
  function Deal(NUMBER, LIMIT : positive) return Random_Sequence;
end APL;

```

End Listing Five A

## LISTING FIVE B

LISTING 5 (Part B) - APL package body

```

-- APLB.ADA
-- 20 JULY 1984
-- DO WHILE JONES

-- This package simulates two APL functions.

-- Note: Roll uses the RND function which returns a random
--   real number between 0.0 and 1.0. The RND function is
--   implementation specific to Maranatha A.

```

```

package body APL is

  function Roll(LIMIT : positive) return positive is
    RANDOM : float;
  begin
    RANDOM := float(LIMIT)*RND(0.0); -- RND is implementation specific.
    return positive(RANDOM+0.5);
  end Roll;

  function Deal(NUMBER, LIMIT : positive) return Random_Sequence is
    MAX : positive := LIMIT;
    RS : Random_Sequence(1..NUMBER);
    SOURCE : Random_Sequence(1..LIMIT);
    RANDOM_INDEX : positive;
  begin
    for i in 1..LIMIT loop
      SOURCE(i) := i; -- SOURCE has one of every number
    end loop;
  end;

```

```

for i in 1..NUMBER loop
  RANDOM_INDEX := Roll(MAX);
  RS(i) := SOURCE(RANDOM_INDEX); -- pick a random number from SOURCE
  for j in RANDOM_INDEX..MAX-1 loop
    SOURCE(j) := SOURCE(j+1); -- remove that number from the SOURCE
  end loop;
  MAX := MAX-1; -- there is now 1 less number in the source array
end loop;
return RS;
end Deal;
end APL;

```

End Listing Five B

## LISTING SIX

LISTING 6 - Complete Draw Poker program

```

-- DPOKER.ADA
-- 19 JULY 1984
-- DO WHILE JONES

with CON_IO; use CON_IO;
with PLAYING_CARDS; use PLAYING_CARDS;
procedure Draw_Poker is

  type Values is (NOTHING, TWO_PAIR, THREE_OF_A_KIND, STRAIGHT,
    FLUSH, FULL_HOUSE, FOUR_OF_A_KIND, STRAIGHT_FLUSH, ROYAL_FLUSH);

  STOCK : Decks;
  PLAYERS_HAND : Hands;
  WAGER, PAYOFF : integer;
  VALUE : Values;

  procedure put(X : Values) is
  begin
    case X is
      when TWO_PAIR => put("Two Pair");
      when THREE_OF_A_KIND => put("Three of a Kind");
      when STRAIGHT => put("a Straight");
      when FLUSH => put("a Flush");
      when FULL_HOUSE => put("a Full House");
      when FOUR_OF_A_KIND => put("Four of a Kind");
      when STRAIGHT_FLUSH => put("a Straight Flush");
      when ROYAL_FLUSH => put("a Royal Flush");
      when NOTHING => put("a losing hand");
    end case;
  end put;

  function Value_of(HAND : Hands) return Values is separate;
  begin
    Open_New(STOCK);
    loop
      put("How many dollars do you want to bet? "); get(WAGER);
      exit when WAGER = 0;
      Shuffle(STOCK);
      Open_New(PLAYERS_HAND);
      for I in 1 .. 5 loop
        Deal_A_Card(PLAYERS_HAND, STOCK);
      end loop;
      put(PLAYERS_HAND);
      Discard_From(PLAYERS_HAND);
    loop
      exit when Filled(PLAYERS_HAND);
      Deal_A_Card(PLAYERS_HAND, STOCK);
    end loop;
    put(PLAYERS_HAND);
    VALUE := Value_of(PLAYERS_HAND);
    case VALUE is
      when ROYAL_FLUSH => PAYOFF := 250;
      when STRAIGHT_FLUSH => PAYOFF := 50;
      when FOUR_OF_A_KIND => PAYOFF := 25;
      when FULL_HOUSE => PAYOFF := 6;
      when FLUSH => PAYOFF := 5;
      when STRAIGHT => PAYOFF := 4;
      when THREE_OF_A_KIND => PAYOFF := 3;
      when TWO_PAIR => PAYOFF := 2;
      when others => PAYOFF := 0;
    end case;
    if PAYOFF = 0
      then put_line("Sorry, you lose.");
      else put("You have "); put(VALUE); put("!");
      new_line;
      put("You win"); put(WAGER*PAYOFF); put_line(" dollars!");
    end if;
  end loop;
  end Draw_Poker;

```

End Listing Six

## LISTING SEVEN

LISTING 7 - Value\_of subprogram

```

-- VALUE.ADA
-- 19 JULY 1984
-- DO WHILE JONES

separate (Draw_Poker); -- real Ada doesn't have a semicolon here
function Value_of(HAND : Hands) return Values is

  PATTERN : String(1..CARDS_IN_HAND-1);
  X : Hands;

  function Flush_in(HAND : Hands) return boolean is

```

(Continued on next page)

Top  
Quality  
Programming  
Tools  
from the  
Developers  
of the  
TurboPower  
Utilities



## Turbo EXTENDER™

Tired of fighting 64K Code and Data Segments?  
Bored while waiting for your 10,000 liner to Compile?  
Want to optimize those sluggish Overlays?

### LARGE CODE MODEL

Write Turbo Pascal programs using all 640K of MSDOS memory, based on any number of separately compiled modules. Provides complete parameter passing using normal Pascal syntax. Heap and Data Segment are shared between all modules. No memory-resident kludges or unnatural parameter passing schemes. Comes with a utility which automatically converts your existing applications.

### LARGE DATA ARRAYS

Transparently access 1 and 2 dimensional arrays of any conceivable size and type. Four models support Normal RAM to 640K, Expanded memory (EMS) to 2Meg, Virtual (Disk-based) to 30Meg, and sparse arrays like the most advanced spreadsheets. Comes with a fast full-screen array browser.

### MAKE FACILITY

A Unix-like MAKE program that is optimized for the Turbo EXTENDER large code model. Rebuild multi-module programs with no wasted effort.

### OVERLAY ANALYST

Perform Static and Dynamic analysis of overlayed Turbo programs. Determine sizes of all procedures in each overlay group. Monitor the running program to find the number of overlay reads, procedure calls, and the load address of all procedures.

### AND EVEN MORE!

DISK CACHE can be incorporated in your program to speed up disk reads for data bases, overlays, etc. Multi-file full screen BROWSE works on any text file. Pascal ENCRYPTOR makes your source safe from prying eyes, improves compile speed 15-30% and leaves the code 100% functional. SHELL generator creates fast compiling shells of unexercised code.

Two DSDD disks with complete Source Code, 100 page printed manual, 30 day guarantee! Requires Turbo Pascal 3.0 and DOS 2.X or 3.X. Runs on IBM PC/XT/AT and compatibles. Call for generic MSDOS support.

**\$85**  
complete

Also get the TurboPower Utilities with the acclaimed Pascal Structure Analyzer

Includes a Pretty Printer, Execution Profiler, and powerful Text and Command Automation Tools.

With full source \$95, executable only \$55.

Credit Card Orders only call Toll-free 7 days per week (US)800-538-8157x830 (CA)800-672-3470x830 PO, COD, Dealers, Questions, Brochures, call or write:

**TURBO**  
Power

478 W. Hamilton #196  
Campbell, CA 95008  
ph. 408-378-3672  
M-F 9AM-5PM PST

Circle no. 207 on reader service card.

# Fatten Your Mac for \$5.00

Thanks to Macintosh owners everywhere, *Dr. Dobb's* January 1985 issue #99 was a runaway best-seller.

Now, due to popular demand, the Doctor has reprinted the sought-after **Fatten Your Mac** article from the sold-out January issue. The article explains how you can pack a full 512K of memory into your system, and save half the cost by performing the upgrade yourself.

To order: Enclose \$5.00 for each copy with this coupon and send to:

*Dr. Dobb's Journal*, 2464 Embarcadero Way, Palo Alto, CA 94303  
Outside U.S., add \$2.00 per copy for shipping & handling.

Please send me \_\_\_\_\_ copies of **Fatten Your Mac. ALL REPRINT ORDERS MUST BE PREPAID.**

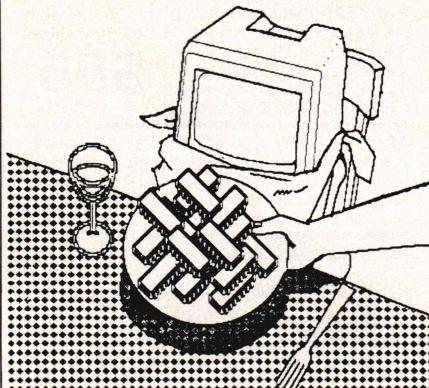
Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Please allow 6-9 weeks for delivery.

3112G



## ADA

### LISTING SEVEN (Listing continued, text begins on page 42)

```

begin
  for i in 1..CARDS_IN_HAND-1 loop
    if Suit_of(Card_Number(i, HAND)) /= Suit_of(Card_Number(i+1, HAND))
      then return FALSE;
    end if;
  end loop;
  return TRUE;
end Flush_in;

function Straight_in(HAND : Hands) return boolean is
begin
  for i in 1..CARDS_IN_HAND-1 loop
    if Ranks'pos(Rank_of(Card_Number(i, HAND)))
      /= Ranks'pos(Rank_of(Card_Number(i+1, HAND)))-1 then
      return FALSE;
    end if;
  end loop;
  return TRUE;
end Straight_in;

begin
  X := HAND; -- make a copy of HAND so it can be sorted
  Sort(X);
  for i in 1..CARDS_IN_HAND-1 loop
    if Rank_of(Card_Number(i, X)) = Rank_of(Card_Number(i+1, X)) then
      PATTERN(i) := 'S'; -- adjacent cards have SAME rank
    else
      PATTERN(i) := 'D'; -- adjacent cards have DIFFERENT rank
    end if;
  end loop;
  if Flush_in(X) and Straight_in(X) then
    if Rank_of(Card_Number(5, X)) = ACE then
      return ROYAL_FLUSH;
    else
      return STRAIGHT_FLUSH;
    end if;
  end if;

  if PATTERN = "SSSD" or PATTERN = "DSSS" then
    return FOUR_OF_A_KIND;
  end if;

  if PATTERN = "SSDS" or PATTERN = "SDSS" then
    return FULL_HOUSE;
  end if;

  if Flush_in(X) then
    return FLUSH;
  end if;

  if Straight_in(X) then
    return STRAIGHT;
  end if;

  if PATTERN = "SSDD" or PATTERN = "DSSD" or PATTERN = "DDSS" then
    return THREE_OF_A_KIND;
  end if;

  if PATTERN = "SDSD" or PATTERN = "DSDS" or PATTERN = "SDDS" then
    return TWO_PAIR;
  end if;

  return NOTHING;
end Value_of;

```

**End Listing Seven**

### LISTING EIGHT

#### LISTING 8 - Corrected Value\_of subprogram

```

-- VALUE2.ADA
-- 9 NOVEMBER 1984
-- DO WHILE JONES

-- This revision recognizes that TWO, THREE, FOUR,
-- FIVE, ACE is a straight (but not a royal flush).

```

```

separate (Draw_Poker); -- real Ada doesn't have a semicolon
function Value_of(HAND : Hands) return Values is

  PATTERN : String(1..CARDS_IN_HAND-1);
  X : Hands;

  function Flush_in(HAND : Hands) return boolean is
  begin
    for i in 1..CARDS_IN_HAND-1 loop
      if Suit_of(Card_Number(i, HAND)) /= Suit_of(Card_Number(i+1, HAND))
        then return FALSE;
      end if;
    end loop;
    return TRUE;
  end Flush_in;

  function Straight_in(HAND : Hands) return boolean is
  begin
    if Rank_of(Card_Number(1, HAND)) = TWO and
      Rank_of(Card_Number(2, HAND)) = THREE and
      Rank_of(Card_Number(3, HAND)) = FOUR and
      Rank_of(Card_Number(4, HAND)) = FIVE and
      Rank_of(Card_Number(5, HAND)) = ACE then
      return TRUE;
    end if;
  end Straight_in;

```

```

for i in 1..CARDS IN HAND-1 loop
  if Ranks'pos(Rank_of(Card_Number(i, HAND)))
    /= Ranks'pos(Rank_of(Card_Number(i+1, HAND)))-1 then
      return FALSE;
    end if;
  end loop;
  return TRUE;
end Straight_in;

begin
  X := HAND; -- make a copy of HAND so it can be sorted
  Sort(X);
  for i in 1..CARDS IN HAND-1 loop
    if Rank of(Card_Number(i, X)) = Rank of(Card_Number(i+1, X)) then
      PATTERN(i) := 'S'; -- adjacent cards have SAME rank
    else
      PATTERN(i) := 'D'; -- adjacent cards have DIFFERENT rank
    end if;
  end loop;
  if Flush in(X) and Straight in(X) then
    if Rank of(Card_Number(4, X)) = KING then
      return ROYAL_FLUSH;
    else
      return STRAIGHT_FLUSH;
    end if;
  end if;

  if PATTERN = "SSSD" or PATTERN = "DSSS" then
    return FOUR_OF_A_KIND;
  end if;

  if PATTERN = "SSDS" or PATTERN = "SDSS" then
    return FULL_HOUSE;
  end if;

  if Flush in(X) then
    return FLUSH;
  end if;

  if Straight in(X) then
    return STRAIGHT;
  end if;

  if PATTERN = "SSDD" or PATTERN = "DSSD" or PATTERN = "DDSS" then
    return THREE_OF_A_KIND;
  end if;

  if PATTERN = "SDSD" or PATTERN = "DSDS" or PATTERN = "SDDS" then
    return TWO_PAIR;
  end if;

  return NOTHING;
end Value_of;

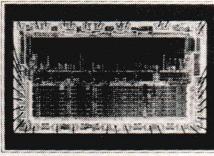
```

**End Listings**

# FROM THE DEVELOPERS OF THE 65816 Microprocessor

*The programming handbook you've been waiting for.*

Programming the  
**65816 Microprocessor**  
Including 6502 and 65C02



- Outlines the programming strengths of the 65816, 6502, and 65C02.
- Includes a review of basic concepts, architecture, and logical operations.

Now available at your local bookstore, or call toll free 1-800-624-0073 to order your copy today. In New Jersey, call 1-800-624-0024.

**Brady**

0-89303-789-3/288 p/\$22.95

Circle no. 219 on reader service card.

## ATTENTION

### C-PROGRAMMERS

#### File System Utility Libraries

All products are written entirely in K&R C. Source code included, No Royalties, Powerful & Portable.

#### BTree Library

**75.00**

- High speed random and sequential access.
- Multiple keys per data file with up to 16 million records per file.
- Duplicate keys, variable length data records.

#### ISAM Driver

**40.00**

- Greatly speeds application development.
- Combines ease of use of database manager with flexibility of programming language.
- Supports multi key files and dynamic index definition.
- Very easy to use.

#### Make

**59.00**

- Patterned after the UNIX utility.
- Works for programs written in every language.
- Full macros, File name expansion and built in rules.

Full Documentation and Example Programs Included.

ALL THREE PRODUCTS FOR —

**149.00**

For more information call or write:

**softfocus**

Credit cards accepted.

1343 Stanbury Drive  
Oakville, Ontario, Canada  
L6L 2J5  
(416) 825-0903

Dealer inquiries invited.

# MODULA-2

## LISTING ONE (Text begins on page 62)

```
MODULE Sorter;
(* Sorter utilizes modules RandomNumbers and Queues to demonstrate *)
(* the utility of data abstraction. *)

FROM InOut IMPORT
    ClearScreen,
    WriteInt,
    WriteLn,
    WriteString;

FROM RandomNumbers IMPORT
    (* PROC *) randu;

FROM Queues IMPORT
    (* PROC *) InitPriorityQueue,
    QueueEmpty,
    Add,
    Fetch,
    (* TYPE *) PriorityQueue;

VAR Count : CARDINAL;
    x : INTEGER;
    Q : PriorityQueue;
BEGIN (* MAIN *)
    ClearScreen;
    InitPriorityQueue(Q); (* Initialize the Priority Queue Q *)
    (* Add 100 pseudo-random numbers to the Priority Queue Q *)
    FOR Count := 1 TO 100 DO
        Add(Q, randu());
    END;
    (* Empty the Priority Queue Q and display each removed element *)
    WriteLn;
    WriteLn;
    WriteString("Sorted Pseudo-Random Numbers.....");
    WriteLn;
    WHILE NOT QueueEmpty(Q) DO
        x := Fetch(Q);
        WriteInt(x, 10);
        WriteLn;
    END;
END Sorter.
```

**End Listing One**

## LISTING TWO

```
DEFINITION MODULE Queues;
EXPORT QUALIFIED
    (* TYPE *) PriorityQueue,
    (* PROC *) InitPriorityQueue,
    Add,
    Fetch,
    QueueEmpty;

TYPE PriorityQueue; (* Opaque Type *)

PROCEDURE InitPriorityQueue(VAR Q : PriorityQueue);
(* Initializes the Priority Queue Q *)

PROCEDURE Add(VAR Q : PriorityQueue; datum : INTEGER);
(* Adds the data item to the Priority Queue Q *)

PROCEDURE Fetch(VAR Q : PriorityQueue) : INTEGER;
(* Fetches the smallest element from the Priority Queue Q *)

PROCEDURE QueueEmpty(Q : PriorityQueue) : BOOLEAN;
(* QueueEmpty RETURNS TRUE if PriorityQueue Q is empty; FALSE otherwise *)

END Queues.

DEFINITION MODULE RandomNumbers;
EXPORT QUALIFIED
    (* PROC *) randu;

PROCEDURE randu() : INTEGER;
(* randu RETURNS a pseud0-random INTEGER *)

END RandomNumbers.
```

**End Listing Two**

## LISTING THREE

```
IMPLEMENTATION MODULE Queues;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;

TYPE PriorityQueue = POINTER TO PriNode;
    PriNode = RECORD
        data : INTEGER;
        link : PriorityQueue;
    END;
```

```

PROCEDURE InitPriorityQueue(VAR Q : PriorityQueue);
BEGIN
  Q := NIL
END InitPriorityQueue;

PROCEDURE Add(VAR Q : PriorityQueue; datum : INTEGER);
VAR T : PriorityQueue;
BEGIN
  IF QueueEmpty(Q) THEN
    NEW(Q);
    Q^.link := NIL;
    Q^.data := datum;
  ELSIF datum < Q^.data THEN
    NEW(T);
    T^.link := Q;
    T^.data := datum;
    Q := T
  ELSE
    Add(Q^.link, datum)
  END;
END Add;

PROCEDURE Fetch(VAR Q : PriorityQueue) : INTEGER;
VAR tempInt : INTEGER;
tempQ : PriorityQueue;
BEGIN
  tempQ := Q;
  tempInt := Q^.data;
  Q := Q^.link;
  DISPOSE(tempQ);
  RETURN tempInt
END Fetch;

PROCEDURE QueueEmpty(Q : PriorityQueue) : BOOLEAN;
BEGIN
  RETURN Q = NIL
END QueueEmpty;

END Queues.

```

```

IMPLEMENTATION MODULE RandomNumbers;
VAR x : INTEGER;
PROCEDURE randu() : INTEGER;
BEGIN
  x := (3 * x + 31) MOD 7;
  RETURN x
END randu;
BEGIN
  x := 7;
END RandomNumbers.

```

### End Listing Three

## LISTING FOUR

```

IMPLEMENTATION MODULE Queues;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
TYPE PriorityQueue = POINTER TO PriNode;
  PriNode = RECORD
    data : INTEGER;
    link : PriorityQueue;
  END;

PROCEDURE InitPriorityQueue(VAR Q : PriorityQueue);
BEGIN
  Q := NIL
END InitPriorityQueue;

PROCEDURE Add(VAR Q : PriorityQueue; datum : INTEGER);
VAR T, T1, NewNode : PriorityQueue;
BEGIN
  IF QueueEmpty(Q) THEN
    NEW(Q);
    Q^.link := NIL;
    Q^.data := datum;
  ELSIF datum < Q^.data THEN
    NEW(T);
    T^.link := Q;
    T^.data := datum;
    Q := T
  ELSE
    T := Q;
    WHILE (T # NIL) & (datum >= T^.data) DO
      T1 := T;
      T := T^.link;
    END;
    NEW(NewNode);
    NewNode^.link := T;
    NewNode^.data := datum;
    T1^.link := NewNode;
  END;
END Add;

PROCEDURE Fetch(VAR Q : PriorityQueue) : INTEGER;
VAR tempInt : INTEGER;
tempQ : PriorityQueue;
BEGIN
  tempQ := Q;
  tempInt := Q^.data;
  Q := Q^.link;
  DISPOSE(tempQ);
  RETURN tempInt
END Fetch;

```

```

PROCEDURE QueueEmpty(Q : PriorityQueue) : BOOLEAN;
BEGIN
  RETURN Q = NIL
END QueueEmpty;

IMPLEMENTATION MODULE RandomNumbers;
VAR x : INTEGER;
PROCEDURE randu() : INTEGER;
BEGIN
  x := (5 * x + 31) MOD 4096;
  RETURN x
END randu;
BEGIN
  x := 7;
END RandomNumbers.

```

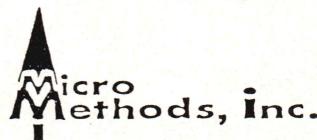
### End Listings

**RP/M2™ creates**  
**CP/M®2.2 compatible**

**IBM PC**

1. Remove the 8088
2. Install the NEC µPD70108
3. Boot PC RP/M2

MSDOS and CP/M 2.2 capability coexist in your IBM PC. The NEC µPD70108 CPU chip is a fast 8088 that can also execute 8080 machine code. PC RP/M2 is standard 2.2 compatible RP/M2 with our CBIOS for the IBM PC. System disk with manual and NEC µPD70108 \$129. Shipping \$5 (\$10 nonUS)  



118 SW First St. - Box G  
 Warrenton, OR 97146  
**(503)861-1765**

# 8080 SIMULATOR

## LISTING TWO (Continued from January)

```

fullcy add.b #$60,rega
        ori #1,ccr
enddaa move sr,regf
        swap regcon0e
        and.w regcon0f,regf
        move.b 0(flagptr,regf.w),regf
        jmp (return)
nofull  tst.b rega
        bra enddaa

nop28  bra illegl ; 28 Illegal for 8080

dadh   move.w regh(regs),d0 ; 29 Dad H
        add.w d0,regh(regs)
        bra docyf

lhld   move.b 1(pseudopc),d0 ; 2A Lhld addr
        rol.w #8,d0
        move.b (pseudopc),d0
        addq.l #2,pseudopc
        move.l d0,a0
        adda.l targbase,a0
        move.b (a0)+,regl(regs)
        move.b (a0),regh(regs)
        jmp (return)

dcxh   dec.w regh(regs) ; 2B Dcx H
        jmp (return)

inrl   inc.b regl(regs) ; 2C Inr L
        move sr,d0
        and.w regcon0e,d0
        and.w regcon01,regf
        or.b 0(flagptr,d0.w),regf
        jmp (return)

dcrl   dec.b regl(regs) ; 2D Dcr L
        move sr,d0
        and.w regcon0e,d0
        and.w regcon01,regf
        or.b 0(flagptr,d0.w),regf
        jmp (return)

mvil   move.b (pseudopc)+,regl(regs) ; 2E Mvi L,nn
        jmp (return)

cma    not.b rega ; 2F Cma
        jmp (return)

nop30  bra illegl ; 30 Illegal for 8080

lxis   move.b 1(pseudopc),d0 ; 31 Lxi S,nnnn
        rol.w #8,d0
        move.b (pseudopc),d0
        addq.l #2,pseudopc
        move.l d0,pseudosp
        adda.l targbase,pseudosp
        jmp (return)

sta    move.b 1(pseudopc),d0 ; 32 Sta addr
        rol.w #8,d0
        move.b (pseudopc),d0
        addq.l #2,pseudopc
        move.b rega,0(targbase,d0.1)
        jmp (return)

inxs   addq.l #1,pseudosp ; 33 Inx S
        jmp (return)

inrm   move.w regh(regs),d0 ; 34 Inr M
        inc.b 0(targbase,d0.1)
        move sr,d0
        and.w regcon0e,d0
        and.w regcon01,regf
        or.b 0(flagptr,d0.w),regf
        jmp (return)

```

```

dcrm  move.w regh(regs),d0          ; 35 Dcr M
      dec.b 0(targbase,d0.l)
      move sr,d0
      and.w regcon0,d0
      and.w regcon01,regf
      or.b 0(flagptr,d0.w),regf
      jmp (return)

mvim  move.w regh(regs),d0          ; 36 Mvi M,nn
      move.b (pseudopc)+,0(targbase,d0.l)
      jmp (return)

stc   bset #0,regf                ; 37 Stc
      jmp (return)

nop38 bra illegl                 ; 38 Illegal for 8080

dads  move.l pseudosp,d0          ; 39 Dad S
      sub.l targbase,d0
      add.w d0,regh(regs)
      bra docyf

lda   move.b 1(pseudopc),d0          ; 3A Lda addr
      rol.w #8,d0
      move.b (pseudopc),d0
      addq.l #2,pseudopc
      move.b 0(targbase,d0.l),rega
      jmp (return)

dcxs  subq.l #1,pseudosp          ; 3B Dcx S
      jmp (return)

inra  move.b rega,regop1(regs)      ; 3C Inr A
      move.b regcon01,regop2(regs)
      move.b regcon0,d0,regop3(regs)
      inc.b rega
      move sr,d0
      and.w regcon0,d0
      and.w regcon01,regf
      or.b 0(flagptr,d0.w),regf

```

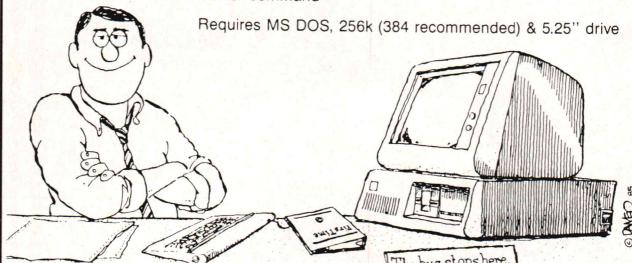
(Continued on next page)

## The First Idea-Processor For Programmers. **FirTime™**

Has features no other editor has.

- Fast program entry through single keystroke statement generators
- Fast editing through syntax oriented cursor movements
- Dramatically reduced debugging time through immediate syntax checking.
- The error checking is thorough and includes semantics • Undefined variables, types and constants • Assignment statements with mismatched types • Errors in include files and macro expansions
- Automatic program formatter (you specify the rules)
- Split Screen editing
- Reading a file with errors moves cursor automatically to point of error
- Unique programmer-oriented features
  - zoom command gives top-down view of program logic
  - view macro command shows expansion of a C macro in the editor
  - view/update include file allows you to view and update an include file
  - transform command allows you to transform statements to related ones
  - search for next error command

Requires MS DOS, 256k (384 recommended) & 5.25" drive



To Order Call: (201) 741-8188 or write:

**SPRUCE TECHNOLOGY CORPORATION**



P.O. Box 7948  
Shrewsbury, NJ 07701

FirstTime for Turbo Pascal	\$ 74.95
FirstTime for dBase III	\$125.00
FirstTime for MS-Pascal	\$245.00
FirstTime for C	\$295.00

FirstTime is a trademark of Spruce Technology Corporation • MS DOS is a trademark of Microsoft Corporation • IBM is a trademark of International Business Machines Inc • Turbo Pascal is a trademark of Borland International • dBase III is a trademark of Ashton Tate

Circle no. 164 on reader service card.

## **COBOL DEVELOPERS**

**PAY LESS FOR THE BEST**

### **TM Level II Cobol** From Micro Focus

Fully Featured, G.S.A., Certified  
Error Free . . . . .

**\$795**

**TM VS Cobol Workbench** Fully integrated package suitable for IBM Mainframe Development. Supporting CICS and IMS Extensions . . . **Call Now**

**Call (713) 342-5875**  
For Free Catalogue

## **The Cobol Shop**

**Cobol Products for Most Environments**  
**Best Prices, Best Support**

TM = Micro Focus Ltd.

Circle no. 247 on reader service card.

# 8080 SIMULATOR

## LISTING TWO (Continued from January)

jmp (return)		jmp (return)	
dcra dec.b rega move sr,d0 and.w regcon0e,d0 and.w regcon01,regf or.b 0(flagptr,d0.w),regf jmp (return)	; 3D Dcr A	moveba move.b rega,regb(regs) jmp (return)	; 47 Mov B,A
mvia move.b (pseudopc)+,rega jmp (return)	; 3E Mvi A,nn	movecb move.b regb(regs),regc(regs) jmp (return)	; 48 Mov C,B
cmc bchq #0,regf jmp (return)	; 3F Cmc	movecc move.b regc(regs),regc(regs) jmp (return)	; 49 Mov C,C
movebb move.b regb(regs),regb(regs) jmp (return)	; 40 Mov B,B	movecd move.b regd(regs),regc(regs) jmp (return)	; 4A Mov C,D
movebc move.b regc(regs),regb(regs) jmp (return)	; 41 Mov B,C	movece move.b rege(regs),regc(regs) jmp (return)	; 4B Mov C,E
movebd move.b regd(regs),regb(regs) jmp (return)	; 42 Mov B,D	movech move.b regh(regs),regc(regs) jmp (return)	; 4C Mov C,H
movebe move.b rege(regs),regb(regs) jmp (return)	; 43 Mov B,E	movecl move.b regl(regs),regc(regs) jmp (return)	; 4D Mov C,L
movebh move.b regh(regs),regb(regs) jmp (return)	; 44 Mov B,H	movecm move.w regh(regs),d0 move.b 0(targbase,d0.1),regc(regs) jmp (return)	; 4E Mov C,M
movebl move.b regl(regs),regb(regs) jmp (return)	; 45 Mov B,L	moveca move.b rega,regc(regs) jmp (return)	; 4F Mov C,A
movebm move.w regh(regs),d0 move.b 0(targbase,d0.1),regb(regs)	; 46 Mov B,M	movedb move.b regb(regs),regd(regs) jmp (return)	; 50 Mov D,B
		movedc move.b regc(regs),regd(regs) jmp (return)	; 51 Mov D,C

Published for Over 7 Years

# 68' MICRO JOURNAL

Serving The 68XXX  
User Worldwide  
The Bible of the  
Serious User

<b>\$2.95 USA</b>
A \$4.75
S \$9.45
M \$9.45
New Zealand NZ \$6.50
H \$23.50
Sweden 30-SEK

**68020**  
**68010**  
**68000**  
**68008**  
**6809**

The ABC's of Languages  
ADA-Basic-C

Subscribe NOW and Receive FREE Reprints Previous (1-8)  
Continuous ADA Series offer exp. 2-15-86

**Hardware-Software-Application Articles**  
**New Product Releases-Announcements**  
**Hints-Kinks-Fixes, etc.**

#### Subscription Rates

U.S.A.: 1 Yr. \$24.50, 2 Yrs. \$42.50, 3 Yrs. \$64.50  
\*Foreign Surface: Add \$12.00 per Year to USA Price  
\*Foreign Airmail: Add \$48.00 per Year to USA Price  
\*Canada & Mexico: Add \$9.50 per Year to USA Price  
\* U.S. Currency or Check or Draft Drawn on USA Bank !!



**2 (615) 842-6809**

Telex 5106006630



5900 Cassandra Smith Rd.

Hixson Tn. 37343

Circle no. 213 on reader service card.

## INDUSTRIAL PASCAL FOR THE 68000

If you're looking for a language to write real-time process control software, look no further. With the rising cost of labor, it is becoming critical that a high level language be used whenever possible. Find out why over 1400 companies have switched to OmegaSoft Pascal for their demanding applications.

OmegaSoft Pascal takes the Pascal framework and expands the basic data types, operators, functions, and memory allocation to fit the needs of real-time systems. These additions fit in the same structure as Pascal and enhance its usefulness without impairing the excellent readability, ease of maintenance, and structured design.

The compiler package includes the compiler, interactive symbolic debugger, relocatable macro assembler, linking loader, and screen editor. Source code is provided for the debugger, screen editor and runtime library.

Versions to run under the OS-9/68000 and VERSAdos operating systems are currently available to end-users and OEM's. End user price is \$900 (domestic) or \$925 (international). A version for CP/M-68K is available for OEM use, with OEM versions for UNIX type operating systems to follow.

T M OmegaSoft is a trademark of Certified Software Corporation. OS-9/68000 is a trademark of Microware. VERSAdos is a trademark of Motorola. CP/M-68K is a trademark of DRI. UNIX is a trademark of Bell Labs.

## CERTIFIED SOFTWARE CORPORATION

616 Camino Caballo, Nipomo, CA 93444  
Telephone: (805) 929-1395; Telex: 467013

Circle no. 209 on reader service card.

movedd move.b regd(regs),regd(regs)	; 52 Mov D,D	moveea move.b rega,rege(regs)	; 5F Mov E,A
movede move.b rege(regs),regd(regs)	; 53 Mov D,E	movehb move.b regb(regs),regh(regs)	; 60 Mov H,B
movedh move.b regh(regs),regd(regs)	; 54 Mov D,H	movehc move.b regc(regs),regh(regs)	; 61 Mov H,C
movedl move.b regl(regs),regd(regs)	; 55 Mov D,L	movehd move.b regd(regs),regh(regs)	; 62 Mov H,D
movedm move.w regh(regs),d0	; 56 Mov D,M	movehe move.b rege(regs),regh(regs)	; 63 Mov H,E
moveb move.b 0(targbase,d0.1),regd(regs)		movehh move.b regh(regs),regh(regs)	; 64 Mov H,H
moveda move.b rega,regd(regs)	; 57 Mov D,A	movehl move.b regl(regs),regh(regs)	; 65 Mov H,L
moveeb move.b regb(regs),rege(regs)	; 58 Mov E,B	movehm move.w regh(regs),d0	; 66 Mov H,M
moveec move.b regc(regs),rege(regs)	; 59 Mov E,C	moveb move.b 0(targbase,d0.1),regh(regs)	
moveed move.b regd(regs),rege(regs)	; 5A Mov E,D	moveha move.b rega,regh(regs)	; 67 Mov H,A
moveee move.b rege(regs),rege(regs)	; 5B Mov E,E	movelb move.b regb(regs),regl(regs)	; 68 Mov L,B
moveeh move.b regh(regs),rege(regs)	; 5C Mov E,H	movelc move.b regc(regs),regl(regs)	; 69 Mov L,C
moveel move.b regl(regs),rege(regs)	; 5D Mov E,L	moveld move.b regd(regs),regl(regs)	; 6A Mov L,D
moveem move.w regh(regs),d0	; 5E Mov E,M	movele move.b rege(regs),regl(regs)	; 6B Mov L,E
moveb move.b 0(targbase,d0.1),rege(regs)			
jmp (return)			

(Continued on page 102)

*Frustrated With the Tyranny of PASCAL? Tired of the Drudgery of BASIC?*

## Free Yourself With CCSM, the Database Language...only \$59.95

*Compare This Routine to Your Present Language, and See the Difference*

```

RD  READ "NAME: ",NAM,! QUIT:NAM="""
IF NAM'?2.A1","1A.E WRITE "PLEASE ENTER AS LAST, FIRST MI",! GO RD
TEL  READ "TEL # ",TEL,! IF TEL'?3N1"--4N WRITE "NNN-NNNN PLEASE",! GO TEL
      SET ^DATA(NAM)=TEL GO RD
PRT  WRITE "      NAME",?20,"TELEPHONE #",! SET NAME="""
LP   SET NAM=$ORDER(^DATA(NAM)) QUIT:NAM="" WRITE NAM,?20,^DATA(NAM),! GO LP

```

This simple program accepts, screens and saves names and phone numbers... sorts and prints them. These six lines of code are an example of the extremely compact, and familiar nature of COMP Computing Standard **MUMPS**, the Database Language. In lines 1 and 2, READ, IF, WRITE and GO should be easy to follow. The pattern match operator "?" filters for the correct input of alpha characters to make a name. In line 4, SET ^ DATA creates a permanent global file, with NAM as a subscript. The data node is SET to the telephone number. In line 6, the \$ORDER command gets the next subscript in order, from the ^ DATA file, thereby SETting NAM to the next name in the file.

CCSM, the Database Language, frees you from the tyranny of typed and restrictive languages... NO declarations of variables or data files. Look at these Features:

- Full Screen Editor
- Virtual Memory (routines and variables may be as large as a disk)
- Multi-User available...up to 15
- B-Tree File Structure
- 8087 and BCD Support
- Exceeds 1984 ANSI Standard **MUMPS**
- Transportable from Micro to Mini to Mainframe

CCSM, the Database Language, is a fast, modern version of ANSI Standard **MUMPS**, developed by COMP Computing. It comes with a 20 year history of development, solving database applications. CCSM improves programmer productivity, and efficiency...typical programs are written in 1/3 the code of BASIC or PASCAL. CCSM is an easy to learn language and comes with a 250 page manual.

AMEX, VISA and  
MASTERCARD  
accepted by phone.

**1-800-257-8052**

In Texas 713-529-2576

CCSM, the Database Language, sells for \$59.95, and comes with full documentation. Until March 31, 1986, for an additional \$15.95, we'll send along the "Cookbook of MUMPS", and its disk, (reg. \$24.95) containing useful routines and utilities. For charts and graphs, order the Graphics disk for \$49.95. Multi-user version, \$450. Disks are non-copy-protected. Requires IBM PC or compatible with 128K. (Macintosh version available...\$89.95)

Order by phone, or clip and mail:

**1-800-257-8052**  
in Texas, 713-529-2576

AMEX	card no.	exp. date
VISA	—	—
MC	—	—

CCSM, Cookbook, and disks special package      **\$75.90**  
CCSM, the Database Language      **\$59.95**  
Graphics disk      **\$49.95**

Please add \$3.00 for shipping and handling. Texas residents add 6 1/8% sales tax.

name \_\_\_\_\_

street \_\_\_\_\_

city \_\_\_\_\_ st \_\_\_\_\_ zip \_\_\_\_\_

*IBM PC and Macintosh are trademarks of International Business Machines, and Apple Computer.*

Circle no. 98 on reader service card.



Dr. Dobb's Journal of  
**Software Tools**  
FOR THE PROFESSIONAL PROGRAMMER

Launching Our Second Decade

Bringing up a 68K Machine from Scratch

PL/68K: Is it C or is it Assembler?

An 8080 Simulator for the 68K

DOS Shell Notes on 80286 Big DOS

**68K**  
10th Anniversary Issue  
ASSEMBLY LANGUAGE

## Dr. Dobb's Stands Apart

Take a good, hard look at the crowded computer magazine market. If you're a serious microcomputerist, one magazine stands apart. **Dr. Dobb's Journal of Software Tools**

**Dr. Dobb's** is not for everyone. It is written by and for expert programmers, and it's the oldest and most technically sophisticated microcomputer publication available. Since 1976, **Dr. Dobb's Journal** has been the unchallenged leading source of software tools for advanced programmers.

With the industry moving forward faster than ever, you need to stay a step ahead. **Dr. Dobb's** sets the pace with:

- regular columns on C, Unix, MS-DOS and 16-bit software;
- algorithms and problem solving;
- lively discussions of fundamental software issues;
- inside information on commercial languages and operating systems;
- tips on advanced programming topics.

The information and valuable code contained in **Dr. Dobb's** makes each issue indispensable for serious computing professionals and enthusiasts. Don't miss a single issue of this valuable resource. Subscribe today. If you aren't fully satisfied, cancel your subscription and keep the first issue as a free sample.

To start your subscription,  
fill out this coupon and return it to:

**Dr. Dobb's Journal of Software Tools**  
P.O. Box 27809, San Diego, CA 92128

• Yes! Please enter my subscription for 12 issues of:  
**Dr. Dobb's Journal of Software Tools** for \$29.97

• If I am not fully satisfied I will write "cancel" on my subscription invoice and keep the first issue as a free sample.

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_

Bill me later  I've enclosed \$27.97 Charge my  VISA

AMERICAN EXPRESS  M/C

Card No. \_\_\_\_\_

Expiration \_\_\_\_\_

Signature \_\_\_\_\_

Offer good in U.S. only Allow 6 to 8 weeks for Delivery 3050

Dr. Dobb's Journal of  
**Software Tools**  
FOR THE PROFESSIONAL PROGRAMMER

# DDJ BACK ISSUES

## #73 Volume VII, Issue 11:

Wildcard UNIX Filenames—Tests for Pidgin—68000 Cross Assembler Listing, Part 2—Adding More BDOS Calls—The Perfect Hash—BASIC Memory Management—Benchmarks for CP/M-86 vs. MSDOS, and the 8087.

## #78 Volume VIII, Issue 4:

RECLAIM Destroyed Directories—Binary Magic Numbers—8080 Fig-Forth Directory & File System—SAY! Forth Votrax Driver—TRS-80 8080 to Z80 Translator—Basic Disk I/O, Part II.

## #79 Volume VIII, Issue 5:

The Augusta Compiler—A Fast Circle Routine—Enhancing the C Screen Editor—Shifts and Rotations on the Z80—The SCB, TSX, and TXS Instructions of the 6502 and 6800—MS-DOS vs. CP/M-86—Controlling MBASIC—The Buffered Keyboard—IBM PC Character Set Linker—Flip Utility for the IBM PC.

## #80 Volume VIII, Issue 6:

Fast Divisibility Algorithms—B-Tree ISAM Concepts—CP/M BDOS and BIOS Calls for C—Serial Expansion in Forth—Fast Matrix Operations in Forth, Part I—Yes, You Can Trace Through BDOS—Julian Dates for Microcomputers—8088 Addressing Modes—8088 Line Generator—CP/M Plus.

## #81 Volume VIII, Issue 7:

The Augusta Compiler, continued—RED: A Better Screen Editor, Part I—Anatomy of a Digital Vector and Curve Generator—Fast Matrix Operations in Forth, Part II—The AGGHHHI Program—MBOOT Revisited—CP/M Plus Feedback—MS-DOS Rebuttal—68000 Tools—Sizing Memory on the IBM PC.

## #82 Volume VIII, Issue 8:

Serial-to Parallel: A Flexible Utility Box—McWORDER: A Tiny Text Editor—And Still More Fifth Generation Computers—Specialist Symbols and I/O Benchmarks for CP/M Plus—CP/M Plus Memory Management—Zero Length File Test—PAUSEIF, QUITIF, and now SKIPIF—ACTxx Cross Assemblers.

## #83 Volume VIII, Issue 9:

FORTH ISSUE: Forth and the Motorola 68000—Nondeterministic Control Words in Forth—A68000 Forth Assembler—GO in Forth—Precompiled Forth Modules—Signed Integer Division—Some Forth Coding Standards—The Forth Sort.

## #84 Volume VIII, Issue 10:

Unix to CP/M Floppy Disk File Conversion—A Small-C Help Facility—Attaching a Winchester Hard Disk to the S-100 Bus—Using Epson Bit-Plot Graphics—8086/88 Function Macros—Auto Disk Format Selection—CP/M Plus Device Tables.

## #85 Volume VIII, Issue 11:

A Kernel for the MC68000—A DML Parser—Towards a More Writable Forth Syntax—Simple Graphics for Printer—Floating-Point Benchmarks.

## #86 Volume VIII, Issue 12:

Faster Circles for Apples—Cursor Control for Dumb Terminals—Dysan's Digital Diagnostic Diskette—Interfacing a Hard Disk Within a CP/M Environment—The New MS-DOS EXEC Function.

## #87 Volume IX, Issue 1:

A structured Preprocessor for MBASIC—A Simple Window Package—Forth to PC-DOS Interface—Sorted Diskette Directory Listing for the IBM PC—Emulate WordStar on TOPS-20—More on optimizing compilers—The PIP mystery device contest.

## #88 Volume IX, Issue 2:

Telecommunications Issue: Micro To Mainframe Connection—Communications Protocols—Unix to Unix Network Utilities—VPC: A Virtual Personal Computer for Networks—PABX on the Personal Computer—BASIC Language Telecommunications Programming—U.S. Robotics S-100 Card Modem.

## #89 Volume IX, Issue 3:

RSA: A Public Key Cryptography System, Part I—Introduction to PL/C: Programming Language for Compilers—Program Design Using Pseudocode—More on Binary Magic Numbers—How fast is CP/M Plus?—CP/M 2.2 BIOS Function: SELDSK—The results of the Floating-Point benchmark.

## #90 Volume IX, Issue 4:

Optimizing Strings in C—Expert Systems and the Weather—RSA: A Public Key Cryptography System, Part II—Several items on CP/M Plus, CP/M 2.2 Compatibility—BDOS Function 10: Vastly improved—More on MS-DOS EXEC Function—Low-Level Input-Output in C.

## #91 Volume IX, Issue 5:

Introduction to Modula-2 for Pascal Programmers—Converting Fig-Forth to Forth-83—Sixth Generation Computers—A New Library for Small-C—Solutions to Quirks in dBASE II.

## #92 Volume IX, Issue 6:

CP/M on the Commodore 64—dBASE II Programming Techniques—First Chinese Forth: A Double-Headed Approach—cc-A Driver for a Small-C Programming System—A New Library for Small-C (Part II)—Comments on Sixth Generation Computers—Review of Turbo Pascal.

## #95 Volume IX, Issue 9:

Forth Special Issue—File Maintenance in Forth—Forth and the Fast Fourier Transform—Computing with Streams—A Forth Native-Code Cross Compiler for the MC68000—The FVG Standard Floating-Point Extension—CP/M Plus: Interbank Memory Moves Without DMA—Ways to make C more powerful and flexible.

## #96 Volume IX, Issue 10:

More dBASE II Programming Techniques—Simple Calculations with Complex Numbers—GREP.C: A Unix-like Generalized Regular Expression Parser—An optimization scheme for compilers, MSDOS 2.0 Filters, Sizing RAM under MSDOS, Two programming systems illustrating Runge-Kutta integration.

## #97 Volume IX, Issue 11:

Adding Primitive I/O Functions to muLISP—Program Monitor Package: Using Interrupts to Instrument Applications—CP/M 2.2 Goes PUBLIC—A Guide to Resources for the C Programmer—RESORT.

## #104 Volume X, Issue 6:

Information Age Issues—Modems: 2400 Bit/Sec and Beyond—C UART Controller—Christensen Protocols in C.

## #105 Volume X, Issue 7:

Build a Custom PC or Clone—The Ultimate Parallel Print Spooler—Designing a Real-Time Clock for the S-100 Bus.

## #106 Volume X, Issue 8:

C Compilers for MSDOS—A Peephole Optimizer for Assembly Language Source Code—Small C Update—Asynchronous Protocols.

## #107 Volume X, Issue 9:

Parallel Pattern Matching and Fgrep—Bose-Nelson Sort—Two T<sub>E</sub>X Implementations for the IBM PC.

## #108 Volume X, Issue 10:

Special Forth Issue—A Threaded-Code Microprocessor Bursts Forth—Design a Forth Target Compiler.

## #109 Volume X, Issue 11:

Modula-2 vs. Pascal for Microcomputers: An Update—Bit Manipulation in Modula-2.

## #110 Volume X, Issue 12:

GEM, Topview and Windows as Programming Environments—Operating System Extensions for CP/M-68K, ProDOS, CP/M—Converting to DOS 3.0.

## #111 Volume XI, Issue 1:

Bringing Up a 68K Machine from Scratch—PL/68K: Is it C or is it Assembler—An 8080 Simulator for the 68K—DOS Shell Notes on 80286 Big DOS.

## TO ORDER:

Return this coupon with payment to: **Dr. Dobb's Journal**, 2464 Embarcadero Way, Palo Alto, CA 94303

Please send me the issue(s) circled: **71 72 73 78 79 80**

**81 82 83 84 85 86 87 88 89 90 91 92 95 96**

**97 104 105 106 107 108 109 110 111**

Price: 1 issue—\$5.00, 2-5 issues—\$4.50 each, 6 or more—\$4.00 each.

(There is a \$10.00 minimum for charge orders.)

I have read the postal instructions and understand that I will not receive my order unless I have sent the correct payment amount.

Please charge my:  Visa  M/C  Amer. Exp.

Card No. \_\_\_\_\_ Exp. Date \_\_\_\_\_

Signature \_\_\_\_\_

I enclose \$ \_\_\_\_\_ (U.S. check or money order).

Outside the U.S., add \$.50 per issue.

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_

Outside U.S., add \$.50 per issue. Price includes shipment by second class or foreign surface mail. Within U.S., allow 9 weeks for delivery. For U.P.S. shipment, add \$1.00 for 1-2 issues and \$.50 per issue thereafter—NO P.O. BOXES. Airmail rates: Canada—add \$1.75 per issue; other foreign add \$3.00 per issue.

Please provide a street address rather than a P.O. Box.

## 8080 SIMULATOR

**LISTING TWO (Continued from January)**

movelh	move.b regh(regs),regl(regs) jmp (return)	; 6C Mov L,H	addc	move.b regc(regs),d0 move.b d0,regop1(regs) move.b rega,regop2(regs) move.b regcon0e,regop3(regs) add.b d0,reg move sr,d0 and.w regcon0f,d0 move.b 0(flagptr,d0.w),regf jmp (return)	; 81 Add C
moveell	move.b regl(regs),regl(regs) jmp (return)	; 6D Mov L,L			
movelm	move.w regh(regs),d0 move.b 0(targbase,d0.1),regl(regs) jmp (return)	; 6E Mov L,M			
moveala	move.b rega,regl(regs) jmp (return)	; 6F Mov L,A	adddd	move.b regd(regs),d0 move.b d0,regop1(regs) move.b rega,regop2(regs) move.b regcon0e,regop3(regs) add.b d0,reg move sr,d0 and.w regcon0f,d0 move.b 0(flagptr,d0.w),regf jmp (return)	; 82 Add D
movemb	move.w regh(regs),d0 move.b regb(regs),0(targbase,d0.1) jmp (return)	; 70 Mov M,B			
movemc	move.w regh(regs),d0 move.b regc(regs),0(targbase,d0.1) jmp (return)	; 71 Mov M,C	adde	move.b rege(regs),d0 move.b d0,regop1(regs) move.b rega,regop2(regs) move.b regcon0e,regop3(regs) add.b d0,reg move sr,d0 and.w regcon0f,d0 move.b 0(flagptr,d0.w),regf jmp (return)	; 83 Add E
movemd	move.w regh(regs),d0 move.b regd(regs),0(targbase,d0.1) jmp (return)	; 72 Mov M,D			
moveme	move.w regh(regs),d0 move.b rege(regs),0(targbase,d0.1) jmp (return)	; 73 Mov M,E			
movemh	move.w regh(regs),d0 move.b regh(regs),0(targbase,d0.1) jmp (return)	; 74 Mov M,H	addhh	move.b regh(regs),d0 move.b d0,regop1(regs) move.b rega,regop2(regs) move.b regcon0e,regop3(regs) add.b d0,reg move sr,d0 and.w regcon0f,d0 move.b 0(flagptr,d0.w),regf jmp (return)	; 84 Add H
moveml	move.w regh(regs),d0 move.b regl(regs),0(targbase,d0.1) jmp (return)	; 75 Mov M,L			
halt	bsr service jmp (return)	; 76 Hlt			
movema	move.w regh(regs),d0 move.b rega,0(targbase,d0.1) jmp (return)	; 77 Mov M,A	addl	move.b regl(regs),d0 move.b d0,regop1(regs) move.b rega,regop2(regs) move.b regcon0e,regop3(regs) add.b d0,reg move sr,d0 and.w regcon0f,d0 move.b 0(flagptr,d0.w),regf jmp (return)	; 85 Add L
moveab	move.b regb(regs),rega jmp (return)	; 78 Mov A,B			
moveac	move.b regc(regs),rega jmp (return)	; 79 Mov A,C			
movead	move.b regd(regs),rega jmp (return)	; 7A Mov A,D	addm	move.w regh(regs),d0 move.b 0(targbase,d0.1),d0 move.b d0,regop1(regs) move.b rega,regop2(regs) move.b regcon0e,regop3(regs) add.b d0,reg move sr,d0 and.w regcon0f,d0 move.b 0(flagptr,d0.w),regf jmp (return)	; 86 Add M
moveae	move.b rege(regs),rega jmp (return)	; 7B Mov A,E			
moveah	move.b regh(regs),rega jmp (return)	; 7C Mov A,H			
moveal	move.b regl(regs),rega jmp (return)	; 7D Mov A,L			
moveam	move.w regh(regs),d0 move.b 0(targbase,d0.1),rega jmp (return)	; 7E Mov A,M			
moveaa	jmp (return)	; 7F Mov A,A	addaa	move.b rega,regop1(regs) move.b rega,regop2(regs) move.b regcon0e,regop3(regs) add.b rega,reg move sr,d0 and.w regcon0f,d0 move.b 0(flagptr,d0.w),regf jmp (return)	; 87 Add A
addb	move.b regb(regs),d0 move.b d0,regop1(regs) move.b rega,regop2(regs) move.b regcon0e,regop3(regs) add.b d0,reg move sr,d0 and.w regcon0f,d0 move.b 0(flagptr,d0.w),regf jmp (return)	; 80 Add B	adcb	move.b regf,regop3(regs) asr.b #1,regf move.b regb(regs),d0 move.b d0,regop1(regs) move.b rega,regop2(regs) moveq #0,d1 addx.b d0,reg jmp (return)	; 88 Adc B

```

move sr,d0
and.w regcon0f,d0
move.b 0(flagptr,d0.w),regf
jmp (return)

adcc move.b regf,regop3(regs) ; 89 Adc C
asr.b #1,regf
move.b regc(regs),d0
move.b d0,regop1(regs)
move.b rega,regop2(regs)
moveq #0,d1
addx.b d0,rega
move sr,d0
and.w regcon0f,d0
move.b 0(flagptr,d0.w),regf
jmp (return)

adcd move.b regf,regop3(regs) ; 8A Adc D
asr.b #1,regf
move.b regd(regs),d0
move.b d0,regop1(regs)
move.b rega,regop2(regs)
moveq #0,d1
addx.b d0,rega
move sr,d0
and.w regcon0f,d0
move.b 0(flagptr,d0.w),regf
jmp (return)

adce move.b regf,regop3(regs) ; 8B Adc E
asr.b #1,regf
move.b rege(regs),d0
move.b d0,regop1(regs)
move.b rega,regop2(regs)
moveq #0,d1
addx.b d0,rega
move sr,d0
and.w regcon0f,d0

```

```

move.b 0(flagptr,d0.w),regf
jmp (return)

adch move.b regf,regop3(regs) ; 8C Adc H
asr.b #1,regf
move.b regh(regs),d0
move.b d0,regop1(regs)
move.b rega,regop2(regs)
moveq #0,d1
addx.b d0,rega
move sr,d0
and.w regcon0f,d0
move.b 0(flagptr,d0.w),regf
jmp (return)

adcl move.b regf,regop3(regs) ; 8D Adc L
asr.b #1,regf
move.b regl(regs),d0
move.b d0,regop1(regs)
move.b rega,regop2(regs)
moveq #0,d1
addx.b d0,rega
move sr,d0
and.w regcon0f,d0
move.b 0(flagptr,d0.w),regf
jmp (return)

adcm move.b regf,regop3(regs) ; 8E Adc M
move.w regh(regs),d0
move.l d0,a0
adda.l targbase,a0
asr.b #1,regf
move.b (a0),d0
move.b d0,regop1(regs)
move.b rega,regop2(regs)
moveq #0,d1
addx.b d0,rega
move sr,d0

```

(Continued on next page)

# INTRODUCING DATALIGHT C

The Datalight C Compiler for MSDOS is a full C with all K&R constructs, including bitfields, plus the version 7 extensions. Other features of the compiler are:

- Produces object files (.obj) so just the MSDOS linker is required.
- Floating point performed with 8087 or automatic software floating.
- Over 100 compact library functions with source.
- Compatible with the Lattice C compiler.
- Runs on IBM-PC, and compatibles, running MSDOS 2.0 or later.
- Complete, easy-to-read users' manual with index.
- Highly optimized code for production quality programs.

**\$60**

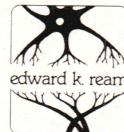
## Datalight

11557 8th Ave. N.E.  
Seattle, Washington 98125  
(206) 367-1803

Outside USA add \$10 shipping. Washington State residents add 7.9% sales tax. VISA and MasterCard accepted.

IBM-PC, a trademark of IBM; MSDOS, a trademark of Microsoft Corp.; Lattice C, a trademark of Lattice Corp.

Circle no. 203 on reader service card.



edward k. ream

Transform Your Programs  
with  
**CPP—C Preprocessor Plus**

Includes ALL features of the standard C preprocessor.■■■■■

- Define arbitrarily complex macros with #define command.
- Include and nest files to any depth with #include command.
- Include lines with #if, #ifdef and #ifndef commands.
- Define multiple constants with #enum command.
- Optional extra feature: Imbed formatting or other commands in your source code. (Lines starting with . or \* are ignored.)

#### Fast and flexible■■■■■

- 30 times faster than the Preprocessor published in Dr. Dobb's Journal.
- Can be used for any language, including assembler.
- Can be used as a stand-alone macro/include processor.
- Code can be used as the lexical analyzer for parsers or assemblers.

#### Complete■■■■■

- You get complete SOURCE CODE in standard C.
- You get everything you need to use CPP immediately.
- CPP is unconditionally guaranteed. If for any reason you are not satisfied with CPP, your money will be refunded promptly.

#### Price: \$95.

Call or write today:

Edward K. Ream  
1850 Summit Ave., Dept. DD  
Madison, WI 53705  
(608) 231-2952

**TO ORDER:** Specify both the operating system (MS-DOS, CP/M 80 or CPM 68K) and the disk format (8 inch CP/M or the exact type of 5 1/4 inch disk). Send a check or money order for \$95 (\$105 for foreign orders). Foreign checks must be denominated in U.S. dollars drawn on a U.S. bank. Sorry, I do NOT accept phone, credit card or COD orders. Please do NOT send purchase orders unless a check is included.

Circle no. 90 on reader service card.

# 8080 SIMULATOR

## LISTING TWO (Continued from January)

and.w regcon0f,d0	and.w regcon0f,d0
move.b 0(flagptr,d0.w),regf	move.b 0(flagptr,d0.w),regf
jmp (return)	jmp (return)
adca move.b regf,regop3(regs)	; 8F Adc A
asr.b #1,regf	sbcc asr.b #1,regf
move.b rega,d0	move.b regc(regs),d0
move.b d0,regop1(regs)	moveq #0,d1
move.b rega,regop2(regs)	subx.b d0,regc
moveq #0,d1	move sr,d0
addx.b d0,regc	and.w regcon0f,d0
move sr,d0	move.b 0(flagptr,d0.w),regf
and.w regcon0f,d0	jmp (return)
move.b 0(flagptr,d0.w),regf	;
jmp (return)	99 Sbb C
subb move.b regb(regs),d0	; 90 Sub B
sub.b d0,regc	sbbd asr.b #1,regf
move sr,d0	move.b regd(regs),d0
and.w regcon0f,d0	moveq #0,d1
move.b 0(flagptr,d0.w),regf	subx.b d0,regd
jmp (return)	move sr,d0
subc move.b regc(regs),d0	; 91 Sub C
sub.b d0,regc	and.w regcon0f,d0
move sr,d0	move.b 0(flagptr,d0.w),regf
and.w regcon0f,d0	jmp (return)
move.b 0(flagptr,d0.w),regf	;
jmp (return)	9B Sbb E
subd move.b regd(regs),d0	; 92 Sub D
sub.b d0,regc	sbbh asr.b #1,regf
move sr,d0	move.b regf(regs),d0
and.w regcon0f,d0	moveq #0,d1
move.b 0(flagptr,d0.w),regf	subx.b d0,regf
jmp (return)	move sr,d0
sube move.b rege(regs),d0	; 93 Sub E
sub.b d0,regc	and.w regcon0f,d0
move sr,d0	move.b 0(flagptr,d0.w),regf
and.w regcon0f,d0	jmp (return)
move.b 0(flagptr,d0.w),regf	;
jmp (return)	9C Sbb H
subh move.b regh(regs),d0	; 94 Sub H
sub.b d0,regc	sbbi asr.b #1,regf
move sr,d0	move.b regl(regs),d0
and.w regcon0f,d0	moveq #0,d1
move.b 0(flagptr,d0.w),regf	subx.b d0,regl
jmp (return)	move sr,d0
subl move.b regl(regs),d0	; 95 Sub L
sub.b d0,regc	and.w regcon0f,d0
move sr,d0	move.b 0(flagptr,d0.w),regf
and.w regcon0f,d0	jmp (return)
move.b 0(flagptr,d0.w),regf	;
jmp (return)	9D Sbb M
subm move.w regh(regs),d0	; 96 Sub M
move.b 0(targbase,d0.l),d0	sbbm move.w regh(regs),d0
sub.b d0,regc	move.l d0,a0
move sr,d0	adda.l targbase,a0
and.w regcon0f,d0	asr.b #1,regf
move.b 0(flagptr,d0.w),regf	move.b (a0),d0
jmp (return)	moveq #0,d1
;	subx.b d0,regc
subm move.w regh(regs),d0	move sr,d0
move.b 0(targbase,d0.l),d0	and.w regcon0f,d0
sub.b d0,regc	move.b 0(flagptr,d0.w),regf
move sr,d0	jmp (return)
and.w regcon0f,d0	;
move.b 0(flagptr,d0.w),regf	9F Sbb A
jmp (return)	;
subaa move.b rega,d0	; 97 Sub A
sub.b d0,regc	sbba asr.b #1,regf
move sr,d0	move.b rega,d0
and.w regcon0f,d0	moveq #0,d1
move.b 0(flagptr,d0.w),regf	subx.b d0,regc
jmp (return)	move sr,d0
;	and.w regcon0f,d0
subbb asr.b #1,regf	move.b 0(flagptr,d0.w),regf
move.b regb(regs),d0	jmp (return)
moveq #0,d1	;
subx.b d0,regc	A0 Ana B
move sr,d0	and.b rega,d0
;	move.b d0,regc
;	and.w regconff,d0
;	move.b 16(flagptr,d0.w),regf
;	jmp (return)
;	;
;	andc move.b regc(regs),d0
;	;

and.b rega,d0 move.b d0,rega and.w regconff,d0 move.b 16(flagptr,d0.w),regf jmp (return)		and.b rega,d0 move.b d0,rega and.w regconff,d0 move.b 16(flagptr,d0.w),regf jmp (return)	
andd move.b regd(regs),d0 and.b rega,d0 move.b d0,rega and.w regconff,d0 move.b 16(flagptr,d0.w),regf jmp (return)	; A2 Ana D	anda move.b rega,d0 and.w regconff,d0 move.b 16(flagptr,d0.w),regf jmp (return)	; A7 Ana A
ande move.b rege(regs),d0 and.b rega,d0 move.b d0,rega and.w regconff,d0 move.b 16(flagptr,d0.w),regf jmp (return)	; A3 Ana E	xrab move.b regb(regs),d0 eor.b d0,rega move.b rega,d0 and.w regconff,d0 move.b 16(flagptr,d0.w),regf jmp (return)	; A8 Xra B
andh move.b regh(regs),d0 and.b rega,d0 move.b d0,rega and.w regconff,d0 move.b 16(flagptr,d0.w),regf jmp (return)	; A4 Ana H	xrac move.b regc(regs),d0 eor.b d0,rega move.b rega,d0 and.w regconff,d0 move.b 16(flagptr,d0.w),regf jmp (return)	; AA Xra C
andl move.b regl(regs),d0 and.b rega,d0 move.b d0,rega and.w regconff,d0 move.b 16(flagptr,d0.w),regf jmp (return)	; A5 Ana L	xrad move.b regd(regs),d0 eor.b d0,rega move.b rega,d0 and.w regconff,d0 move.b 16(flagptr,d0.w),regf jmp (return)	; AA Xra D
andm move.w regh(regs),d0 move.b 0(targbase,d0.1),d0	; A6 Ana M	xrae move.b rege(regs),d0 eor.b d0,rega move.b rega,d0 and.w regconff,d0	; AB Xra E

(Continued on next page)



## LISP

The preferred symbolic processing language  
of the Artificial Intelligence Community

### catch the next micro-wave with **UO-LISP**

Not "just another pretty dialect" but the most powerful implementation of LISP available in the micro market place. For the professional engineers, researchers, and educators, UO-LISP maintains the power and flexibility inherent in LISP while providing the expected functionality of mainframe LISP systems. (+) **UO-LISP steps beyond the competition and provides a real source to native code compiler.**

CPU Family	Operating System	Production System	Learn System	Production plus Learn System
8086	MS-DOS	150 <sup>00</sup>	85 <sup>00</sup>	185 <sup>00</sup>
	PC-DOS	150 <sup>00</sup>	85 <sup>00</sup>	185 <sup>00</sup>
	CPM/86	available soon	—	—
Z80	CPM	125 <sup>00</sup>	85 <sup>00</sup>	160 <sup>00</sup>
	TRS DOS	80 <sup>00</sup>	N/A	N/A

For MORE DETAIL AND TO ORDER:  
Send for *FREE* brochures and order forms.

### NORTHWEST COMPUTER ALGORITHMS

P.O. Box 1747, Novato, CA 94948  
(415) 897-1302

Circle no. 191 on reader service card.

# 8080 SIMULATOR

## LISTING TWO (Continued from January)

```

move.b 16(flagptr,d0.w),regf
jmp (return)

xrah move.b regh(regs),d0 ; AC Xra H
eor.b d0,rega
move.b rega,d0
and.w regconff,d0
move.b 16(flagptr,d0.w),regf
jmp (return)

xral move.b regl(regs),d0 ; AD Xra L
eor.b d0,rega
move.b rega,d0
and.w regconff,d0
move.b 16(flagptr,d0.w),regf
jmp (return)

xram move.w regh(regs),d0 ; AE Xra M
move.b 0(targbase,d0.l),d0
eor.b d0,rega
move.b rega,d0
and.w regconff,d0
move.b 16(flagptr,d0.w),regf
jmp (return)

xraa moveq #0,rega ; AF Xra A
move.b 16(flagptr),regf
jmp (return)

orab move.b regb(regs),d0 ; B0 Ora B
or.b rega,d0
move.b d0,rega
and.w regconff,d0
move.b 16(flagptr,d0.w),regf
jmp (return)

```

```

orac move.b regc(regs),d0 ; B1 Ora C
or.b rega,d0
move.b d0,rega
and.w regconff,d0
move.b 16(flagptr,d0.w),regf
jmp (return)

orad move.b regd(regs),d0 ; B2 Ora D
or.b rega,d0
move.b d0,rega
and.w regconff,d0
move.b 16(flagptr,d0.w),regf
jmp (return)

orae move.b rege(regs),d0 ; B3 Ora E
or.b rega,d0
move.b d0,rega
and.w regconff,d0
move.b 16(flagptr,d0.w),regf
jmp (return)

orah move.b regh(regs),d0 ; B4 Ora H
or.b rega,d0
move.b d0,rega
and.w regconff,d0
move.b 16(flagptr,d0.w),regf
jmp (return)

oral move.b regl(regs),d0 ; B5 Ora L
or.b rega,d0
move.b d0,rega
and.w regconff,d0
move.b 16(flagptr,d0.w),regf
jmp (return)

oram move.w regh(regs),d0 ; B6 Ora M
move.b 0(targbase,d0.l),d0
or.b rega,d0
move.b d0,rega
and.w regconff,d0
move.b 16(flagptr,d0.w),regf
jmp (return)

oraa move.b rega,d0 ; B7 Ora A
and.w regconff,d0
move.b 16(flagptr,d0.w),regf
jmp (return)

cmpb cmp.b regb(regs),rega ; B8 Cmp B
move sr,d0
and.w regconff,d0
move.b 0(flagptr,d0.w),regf
jmp (return)

cmpc cmp.b regc(regs),rega ; BB Cmp C
move sr,d0
and.w regconff,d0
move.b 0(flagptr,d0.w),regf
jmp (return)

cmpd cmp.b regd(regs),rega ; BA Cmp D
move sr,d0
and.w regconff,d0
move.b 0(flagptr,d0.w),regf
jmp (return)

cmpe cmp.b rege(regs),rega ; BB Cmp E
move sr,d0
and.w regconff,d0
move.b 0(flagptr,d0.w),regf
jmp (return)

cmph cmp.b regh(regs),rega ; BC Cmp H
move sr,d0
and.w regconff,d0
move.b 0(flagptr,d0.w),regf
jmp (return)

```

## MODULA-2

### DEVELOPMENT BUILDING BLOCKS

**REPERTOIRE**, from PMI. High-performance tools. Screen display system. Multi-window editor.

#### Screen System.

REPERTOIRE won't bloat your programs because it doesn't generate code. You design screens with any editor, then REPERTOIRE's screen compiler transforms them into one dense, rapid-access file. Your program retrieves and displays screens with a single function call. Screens check user input, control program branching, and give context-sensitive help. REPERTOIRE supports attributes, multiple windows, and any number of screens.

#### Editor.

REPERTOIRE's editor fits neatly into your programs. It can edit multiple files, as big as your memory limit, in user-located windows.

#### High-Performance Low level Routines.

REPERTOIRE provides extensive DOS and BIOS access, string tools (including English-language analyzer), list handling, and a sample directory manager.

Excellent for educational software or any other screen-intensive application. For IBM compatibles; Logitech & ITC versions. Manual and two 360K diskettes of fully commented source code for \$64. Preview documentation for FREE. No royalties. Immediate shipment. Check/MC/VISA.

**PMI** 4536 S.E. 50th Portland, OR 97206 (503) 293-7706  
MCI Mail: 269-1013; Compuserve: 74706,262

Circle no. 239 on reader service card.

cmpl	cmp.b regl(regs),rega move sr,d0 and.w regcon0f,d0 move.b 0(flagptr,d0.w),regf jmp (return)	; BD Cmp L	bne mloop lea.1 0(targbase,d0.1),pseudopc jmp (return)
cmpam	move.w regh(regs),d0 move.l d0,a0 adda.l targbase,a0 cmp.b (a0),rega move sr,d0 and.w regcon0f,d0 move.b 0(flagptr,d0.w),regf jmp (return)	; BE Cmp M	jmpa move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudopc lea.1 0(targbase,d0.1),pseudopc jmp (return)
cmpaa	cmp.b rega,rega move sr,d0 and.w regcon0f,d0 move.b 0(flagptr,d0.w),regf jmp (return)	; BF Cmp A	cnz move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudopc btst #6,regf bne mloop move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp lea.1 0(targbase,d0.1),pseudopc jmp (return)
rnz	btst #6,regf bne mloop	; C0 Rnz	
ret	move.b 1(pseudosp),d0 rol.w #8,d0 move.b (pseudosp),d0 addq.l #2,pseudosp lea.1 0(targbase,d0.1),pseudopc jmp (return)	; C9 Ret	pushb move.b regb(regs),-(pseudosp) move.b regc(regs),-(pseudosp) jmp (return)
popb	move.b (pseudosp)+,regc(regs) move.b (pseudosp)+,regb(regs) jmp (return)	; C1 Pop B	adi move.b (pseudopc)+,d0 move.b d0,regop1(regs) move.b rega,regop2(regs) move.b regcon0e,regop3(regs) add.b d0,rega move sr,d0 and.w regcon0f,d0 move.b 0(flagptr,d0.w),regf jmp (return)
jnz	move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudopc btst #6,regf	; C2 Jnz addr	

(Continued on next page)

# Wizard C

Discover the powers of Wizard C  
for yourself!

"...written by someone who has been in the business a while. This especially shows in the documentation."

Computer Language  
February, 1985

" Wizard's got the highest marks for support."

"The Wizard compiler had excellent diagnostics; it would be easier writing portable code with it than with any other compiler we tested."

Dr. Dobb's Journal  
August, 1985

Full Lint checking, six memory models, 8087 support, in-line assembly language, ROMable data support, full library source code. Cross-compilers are available on VAX/VMS and UNIX machines.

(617) 641-2379

Only \$450.

11 Willow Court  
Arlington, MA 02174



**WIZARD**  
SYSTEMS SOFTWARE, INC.

Circle no. 116 on reader service card.

Now available with  
8087 Support!

**MTBASIC**  
Basic Compiler

#### Features:

Multi-line functions Multitasking

No runtime fee Windowing

Handles interrupts Interactive

Fast native code Compiles in seconds

MTBASIC is easy to use since you can write programs in an interactive environment and then compile them using only one command. MTBASIC has many advanced features like multitasking, random file access, formatted I/O, assembly language calls, and ROMable code.

The MTBASIC package includes all the necessary software to run in interpreter or compiler mode, an installation program (so any system can use windows), demonstration programs, and a comprehensive manual.

#### Ordering

MTBASIC is available for CP/M, MS-DOS, and PC-DOS systems for \$49.95. MTBASIC with 8087 support is available for MS-DOS for \$79.95. Shipping is \$3.50 (\$10.00 overseas). MD residents add 5% sales tax. MC, Visa, checks and COD accepted.

**SOFIAID, Inc.**

P.O. Box 2412 Columbia, MD 21045-1412  
301/792-8096

Circle no. 88 on reader service card.

# 8080 SIMULATOR

## LISTING TWO (Continued from January)

rst0	move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp move.l targbase,pseudopc jmp (return)	; C7 Rst 0	btst #6,regf beq mloop move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp lea.l 0(targbase,d0.1),pseudopc jmp (return)
rz	btst #6,regf beq mloop move.b 1(pseudosp),d0 rol.w #8,d0 move.b (pseudosp),d0 addq.l #2,pseudosp lea.l 0(targbase,d0.1),pseudopc jmp (return)	; C8 Rz	call move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudosp move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp lea.l 0(targbase,d0.1),pseudopc jmp (return)
jz	move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudosp btst #6,regf beq mloop lea.l 0(targbase,d0.1),pseudopc jmp (return)	; CA Jz addr	call move.b regf,regop3(regs) asr.b #1,regf move.b (pseudopc)+,d0 move.b d0,regop1(regs) move.b rega,regop2(regs) moveq #0,d1 addx.b d0,rega move sr,d0 and.w regcon0f,d0
nopCB	bra illegl	; CB Illegal for 8080	aci move.b regf,regop3(regs) asr.b #1,regf move.b (pseudopc)+,d0 move.b d0,regop1(regs) move.b rega,regop2(regs) moveq #0,d1 addx.b d0,rega move sr,d0 and.w regcon0f,d0
cz	move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudosp	; CC Cz addr	

## ¿C? ¡Sí!

If you're a C programmer (or want to be one), we speak your language. Subscribe to **The C Journal** today, and start increasing your productivity right away. We give you information you can use on any machine — IBM PC™, UNIX™-based, Macintosh™, or CP/M™ — micro, mini, or mainframe.

- in-depth reviews and feature articles — C compilers, editors, interpreters, function libraries, and books.
- hints and tips — help you work **better** and **faster**.
- interviews — with software entrepreneurs that **made it** — by using C.
- news and rumors — from the ANSI standards committee and the industry.

### Limited Time Offer

Join our thousands of subscribers at the **Discount Rate** of only \$18 for a full year (regularly \$28)! Call us now at (201) 989-0570 for faster service — don't miss a single issue of **The C Journal**!

Please add \$9 for overseas airmail.

Trademarks — CP/M: Digital Research Inc. IBM PC: IBM Corp. Macintosh: Apple Computer Corp. **The C Journal**: InfoPro Systems. UNIX: AT&T Bell Labs.



**InfoPro Systems**  
3108 Route 10  
Denville, NJ 07834  
(201) 989-0570



Circle no. 194 on reader service card.

## FTL Modula-II \$49.95!



Your next computer language. The successor to Pascal, Modula is powerful. Why? Once a routine is written, it need never be recompiled. Programs work everywhere from Z80 through VAX.

FTL Modula-II is a full Z80 CP/M compiler (MSDOS version soon)! It's **fast** — 18K source compiles in 7 seconds! The built-in split screen editor is worth \$60 alone. Some standard features: full recursion, 15 digit reals, CP/M calls, coprocessors, assembler and linker. The one-pass compiler makes true Z80.COM, ROMable code, too. Get the language you've waited for now. Only \$49.95!

## FTL Editor Toolkit

Full source to our split-screen programming editor. Curious? Want to customize to your tastes? Want sample Modula-II code? This is perfect for you. Comes with all you need for your personal editor or terminal installer. Just \$39.95!

**Workman and Associates**  
112 Marion Avenue  
Pasadena, CA 91106  
(818) 796-4401

We have over 200 formats in stock! Please specify your format when ordering. Add \$2.50 per order for shipping. We welcome COD orders!

Circle no. 244 on reader service card.

move.b 0(flagptr,d0.w),regf jmp (return)			beq outspec cmp.b #55,d0 beq outspec endc
rst8	move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp lea.l \$8(targbase),pseudopc jmp (return)	; CF Rst 8	move.l #\$ff0000,a0 move.b rega,0(a0,d0.1) jmp (return)
rnc	btst #0,regf bne mloop move.b 1(pseudosp),d0 rol.w #8,d0 move.b (pseudosp),d0 addq.l #2,pseudosp lea.l 0(targbase,d0.1),pseudopc jmp (return)	; D0 Rnc	cnc move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudosp btst #0,regf bne mloop move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp lea.l 0(targbase,d0.1),pseudopc jmp (return)
popd	move.b (pseudosp)+,rege(regs) move.b (pseudosp)+,regd(regs) jmp (return)	; D1 Pop D	move.b regd(regs),-(pseudosp) move.b rege(regs),-(pseudosp) jmp (return)
jnc	move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudosp btst #0,regf bne mloop lea.l 0(targbase,d0.1),pseudopc jmp (return)	; D2 Jnc addr	pushd move.b (pseudopc)+,d0 sub.b d0,reg move sr,d0 and.w regcon0f,d0 move.b 0(flagptr,d0.w),regf jmp (return)
out	moveq #0,d0 move.b (pseudopc)+,d0  ifne diskio cmp.b #54,d0	; D3 Out nn	sui move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1
			rst10 ; D7 Rst 10

(Continued on next page)

## Now available for the computer experimenter!

### COMPUTER CONNOISSEUR'S DELIGHT!

NOW BE IN CONTROL WITH YOUR COMPUTER — THE ONLY PUBLICATION OF ITS KIND WRITTEN FOR THE USER. DISCOVER THE SECRETS AND LEARN THE VERSATILITY OF MODERN COMPUTER COMMAND CONTROL CONCEPTS. EXPERIMENT WITH COMPUTER AND TELEPHONE SYSTEMS, INTERFACE THEM, LEARN HOW THEY WORK, WHAT THEY DO... AND HOW TO GET THEM TO WORK FOR YOU! A COMPLETE TELEPHONE ENGINEERING COURSE IS INCLUDED IN MONTHLY CHAPTERS, BRINGING YOU THROUGH STEP, CROSSBAR, ESS, BUBBLE, AND ATOMIC SWITCHING SYSTEMS! EXCLUSIVE COVERAGE IN BIOLOGICAL COMPUTING SYSTEMS, TOO! COMPUTERS AND TELEPHONES ARE THE FUTURE. THIS PUBLICATION IS AN ABSOLUTE MUST FOR EVERYONE INTERESTED.

UNPUBLISHED MATERIAL  
WIT  
COMICS  
DIRECTORY LISTING NET-NETWORKS  
AC-CESS CODES

*The one you've all been waiting for*

NOW AVAILABLE — Learn how to repair telephones and telephone systems, how they work, in monthly installments with the magazine for you.

**Computel™**

PUBLISHED MONTHLY

ONE YEAR SUBSCRIPTION \$14.00  
(SAMPLE COPY \$2.00)  
SUBSCRIPTION & 2 PROGRAMS \$20.00

COMPUTEL—the complete SOURCE for everyone. You can now do the things you've only heard about, right in the privacy of your own home. Indispensable reference to phreaks and hackers. Learn how to get all kinds of computer programs FREE. Get the inside story of big business systems—their quirks and flaws—and remain up to date with vital occurrences within the computer industry. Computel is a publication designed for everyone who has an intense curiosity of computer systems, containing a wealth of hard to find information, codes, and numbers. Published monthly.

**Computel Publishing Society**

6354 VAN NUYS BL., #161-A / VAN NUYS, CA 91401

Circle no. 225 on reader service card.

## QUICK REF

Indexing at your fingertips! Take the pressure off the tedious search for that much needed article.

Quick Ref offers rapid recall of magazine articles by title, author, and key combinations.

FOR THE INTRODUCTORY PRICE OF

**\$34.95**

### QUICK REF

PROVIDES:

- \* Key access to magazine articles
- \* Rapid search by title, author, and key combinations
- \* Convenient on-line help
- \* Easy to use manual

### FREE INTRODUCTORY OFFER WITH PURCHASE

The completely indexed  
Dr. Dobb's Journal  
Data Base

Available for IBM Compatibles and Victor 9000

Terra Base Software  
906 S. 8th Street  
Laramie, Wyoming 82070  
(800) 238-4790 9:00 A.M.—5:00 P.M. M.S.T.

All orders shipped U.P.S. Surface shipping included in price. Visa/MasterCard accepted. Foreign orders please add \$15.00. Checks must be on U.S. Bank in U.S. dollars. Specify machine and DOS version.

Circle no. 231 on reader service card.

# 8080 SIMULATOR

## LISTING TWO (Continued from January)

move.b d1,-1(pseudosp)		addq.l #2,pseudopc	
subq.l #2,pseudosp		btst #0,regf	
lea.l \$10(targbase),pseudopc		beq mloop	
jmp (return)		move.l pseudopc,d1	
rc btst #0,regf	; D8 Rc	sub.l targbase,d1	
beq mloop		move.b d1,-2(pseudosp)	
move.b 1(pseudosp),d0		rol.w #8,d1	
rol.w #8,d0		move.b d1,-1(pseudosp)	
move.b (pseudosp),d0		subq.l #2,pseudosp	
addq.l #2,pseudosp		lea.l 0(targbase,d0.1),pseudopc	
lea.l 0(targbase,d0.1),pseudopc		jmp (return)	
jmp (return)			
nopD9 bra illegl	; D9 Illegal for 8080	nopDD bra illegl	; DD Illegal for 8080
jc move.b 1(pseudopc),d0	; DA Jc addr	sbi asr.b #1,regf	; DE Sbi nn
rol.w #8,d0		move.b (pseudopc)+,d0	
move.b (pseudopc),d0		moveq #0,d1	
addq.l #2,pseudopc		subx.b d0,regf	
btst #0,regf		move sr,d0	
beq mloop		and.w regcon0f,d0	
lea.l 0(targbase,d0.1),pseudopc		move.b 0(flagptr,d0.w),regf	
jmp (return)		jmp (return)	
in moveq #0,d0	; DB In nn	rst18 move.l pseudopc,d1	; DF Rst 18
move.b (pseudopc)+,d0		sub.l targbase,d1	
move.l #\$ff0000,a0		move.b d1,-2(pseudosp)	
move.b 0(a0,d0.1),regf		rol.w #8,d1	
jmp (return)		move.b d1,-1(pseudosp)	
cc move.b 1(pseudopc),d0	; DC Cc addr	subq.l #2,pseudosp	
rol.w #8,d0		lea.l \$18(targbase),pseudopc	
move.b (pseudopc),d0		jmp (return)	
		rpo btst #2,regf	; E0 Rpo

# TheMax™

Want unparalleled speed and efficiency from your Macintosh? Get TheMax.

**TheMax:** a 1.5Mb memory board designed to expand to a full 4Mb of power, *plus*

**MaxRAM:** software to configure your memory two ways: 1Mb of contiguous memory with 400K RAM disk or a 512K Mac with a recoverable 1024K RAM disk, *and*

**MaxPrint:** print spooler software that lets you work and print at the same time.

TheMax is available now for both the 128K and the 512K Macintosh. Kits and 512K upgrades are also available. Ask your dealer for more information, or contact

# 1.5Mb

# MacMemory Inc.

473 MACARA AVE., SUITE 701, SUNNYVALE, CA 94086.  
(408) 773-9922.

Macintosh is a trademark licensed to Apple Computer Inc.

Circle no. 218 on reader service card.

# MASTER•KEY

New MS-DOS Disassembler

## MASTER•KEY

is an intelligent MS-DOS Disassembler. It instantly and automatically produces easily readable self-documented assembly language segmented source files that can be edited and re-assembled from any executable file (COM or EXE). You may immediately scan the listing online if desired.

## MASTER•KEY

includes a Pretty-printed Source Code Formatter and Cross-Reference Generator. The Formatter produces a fully-documented assembly language source listing identifying all branch addresses, ROM BIOS and DOS Functions and Interrupts. It serves as an object code optimizer by helping to identify stack areas, temporary storage, ASCII strings, and unnecessary code. The Cross-Reference identifies the number and location of all branch addresses, labels, symbols, functions, and interrupts.

### Minimum System Requirements:

256K 8088/8086/80186/80286 PC (close to IBM compatibility)  
MS-DOS 2.0, 2.1, 3.0, or 3.1  
One 360K DSDD Floppy Drive (IBM PC Format)

### Sharpe Systems Corporation

2320 "E" Street  
La Verne, CA 91750

**(714) 596-0070**

(MC and Visa accepted)

**\$49.95**  
plus \$3.00 shipping  
and handling  
California residents  
add 6 1/2% sales tax

Circle no. 241 on reader service card.

```

bne mloop
move.b 1(pseudosp),d0
rol.w #8,d0
move.b (pseudosp),d0
addq.l #2,pseudosp
lea.l 0(targbase,d0.1),pseudopc
jmp (return)

poph move.b (pseudosp)+,regl(regs) ; E1 Pop H
move.b (pseudosp)+,regh(regs)
jmp (return)

jpo move.b 1(pseudopc),d0 ; E2 Jpo addr
rol.w #8,d0
move.b (pseudopc),d0
addq.l #2,pseudopc
btst #2,regf
bne mloop
lea.l 0(targbase,d0.1),pseudopc
jmp (return)

xthl move.b regl(regs),d0 ; E3 Xthl
move.b (pseudosp),regl(regs)
move.b d0,(pseudosp)
move.b regh(regs),d0
move.b 1(pseudosp),regh(regs)
move.b d0,(pseudosp)
jmp (return)

cpo move.b 1(pseudopc),d0 ; E4 Cpo addr
rol.w #8,d0
move.b (pseudopc),d0
addq.l #2,pseudopc
btst #2,regf
bne mloop
move.l pseudopc,d1
sub.l targbase,d1
move.b d1,-2(pseudosp)

```

```

rol.w #8,d1
move.b d1,-1(pseudosp)
subq.l #2,pseudosp
lea.l 0(targbase,d0.1),pseudopc
jmp (return)

pushh move.b regh(regs),-(pseudosp) ; E5 Push H
move.b regl(regs),-(pseudosp)
jmp (return)

ani and.b (pseudopc)+,rega ; E6 Ani nn
move.b rega,d0
and.w regconff,d0
move.b 16(flagptr,d0.w),regf
jmp (return)

rst20 move.l pseudopc,d1 ; E7 Rst 20
sub.l targbase,d1
move.b d1,-2(pseudosp)
rol.w #8,d1
move.b d1,-1(pseudosp)
subq.l #2,pseudosp
lea.l $20(targbase),pseudopc
jmp (return)

rpe btst #2,regf ; E8 Rpe
beq mloop
move.b 1(pseudosp),d0
rol.w #8,d0
move.b (pseudosp),d0
addq.l #2,pseudosp
lea.l 0(targbase,d0.1),pseudopc
jmp (return)

```

(To be continued in March)

# Time and Money.

We've just done something we know you'll like. We've made the SemiDisk far more affordable than ever before. With price cuts over 25% for most of our product line. Even our new 2 megabyte units are included.

## It's Expandable

SemiDisk Systems builds fast disk emulators for more microcomputers than anyone else. S-100, IBM-PC, Epson QX-10, TRS-80 Models II, 12, and 16. You can start with as little as 512K bytes, and later upgrade to 2 megabytes per board...at your own pace, as your needs expand. Up to 8 megabytes per computer, using only four bus slots, max! Software drivers are available for CP/M 80, MS-DOS, ZDOS, TurboDOS, VALDOCS 2, and Cromix. SemiDisk turns good computers into great computers.

# SEMI DISK

SemiDisk Systems, Inc., P.O. Box GG, Beaverton, Oregon 97075

## Battery Backup, Too

At 0.7 amps per 2 megabytes, SemiDisk consumes far less power than the competition. And you don't have to worry if the lights go out. The battery backup option gives you 5-10 hours of data protection during a blackout. Nobody else has this important feature. Why risk valuable data?

## The Best News

	<u>512K</u>	<u>1Mbyte</u>	<u>2Mbyte</u>
SemiDisk I, S-100	\$695	\$1395	
SemiDisk II, S-100	\$995		\$1995
IBM PC, XT, AT	\$595		\$1795
QX-10	\$595		\$1795
TRS-80 II, 12, 16	\$695		\$1795
Battery Backup Unit	\$150	\$150	\$150

Someday you'll get a SemiDisk.  
Until then, you'll just have to....wait.



Call 503-646-5510 for CBBS/NW, 503-775-4838 for CBBS/PCS, and 503-649-8327 for CBBS/Aloha. All SemiDisk equipped computer bulletin boards (300/1200 baud) SemiDisk, SemiSpool trademarks of SemiDisk Systems.

Circle no. 85 on reader service card.

## LISTING ONE (Text begins on page 114)

```

;***** PLOTDOT.OBJ Library module for Microsoft C (small model programs)
; by Dan Rollins
;
; Mid-resolution graphics pixel-plot function.
; Uses look-up tables for fastest operation.
; Permission is granted to use this for any purpose whatsoever.
;
; synopsis:
; plotdot(x,y,color)
;   int x;           horizontal (0-319) not value-checked
;   int y;           vertical (0-199) not value-checked
;   int color;       color for dot (0 to 3)

----- preamble for placing data into Microsoft C 'S model' static data area
dgroup group data
data segment word public 'data'

----- lookup table for start of each graphics line
----- index is: (Y * 2)
row_tbl label word
addr = 0
rept 100
  dw addr,addr+2000H ;Y=0,1, 2,3, 4,5, etc
  addr = addr+80
endm

----- lookup table for mid-res pixel positions in relevant byte
----- index is: (X mod 4)
mask_tbl db 0011111b, 11001111b, 11110011b, 11111100b

----- lookup table for color, according to position in byte
----- index is (COLOR * 4) + (X mod 4)
color_tbl db 0000000b, 0000000b, 0000000b, 0000000b ;color 0
           db 0100000b, 0001000b, 00000100b, 0000001b ;color 1
           db 1000000b, 0010000b, 00001000b, 0000010b ;color 2
           db 1100000b, 0011000b, 00001100b, 00000011b ;color 3
data ends

----- preamble for placing code into Microsoft C 'S model' program area
pgroup group prog
prog segment byte public 'prog'
assume cs:pgroup, ds:dgroup

public plotdot
plotdot proc near      ;SMALL MODEL ONLY
  pop  si           ;fetch return addr /this technique is faster
  pop  bx           ;fetch X ordinate /than stack-relative access
  pop  di           ;fetch Y ordinate
  pop  cx           ;fetch color

  mov  dx,es         ;save current ES
  mov  ax,0b800H
  mov  es,ax         ;get set to write to video buffer

  shl  di,1          ;index into row address lookup table

  mov  di, row_tbl[di] ;DI points to start of selected row

  mov  ax,bx         ;copy the X ordinate
  shr  ax,1
  shr  ax,1          ;divided by 4 is byte offset from start of row

  add  di,ax         ;DI points to byte to modify
  mov  al,es:[di]    ;fetch current screen byte

  and  bx,3          ;get pixel-offset in byte (0,1,2, or 3)
  and  al,mask_tbl[bx] ;mask a "hole" into the current byte

----- index into the color table
  and  cx,3          ;make sure it's a valid color
  shl  cx,1
  shl  cx,1          ; COLOR * 2
  add  bx,cx         ; index is (COLOR * 4) + (X mod 4)
  or   al,color_tbl[bx] ;fill the "hole" with selected color

  mov  es:[di],al    ;place modified byte back into screen
  mov  es,dx
  jmp  si           ;artificial (quick) NEAR return to caller
plotdot endp
prog ends
end

```

End Listing One

## LISTING TWO

```

/* **** LINE.C ****
by Dan Rollins
Uses incremental algorithm and fast ASM plotdot
Permission is granted to use this for any purpose whatsoever.
**** */

main(argv) /* this just skips the 'main' function altogether */
char *argv; /* points to the DOS command line */
{
  int x,y;
  dmode(1); /* enter text mode first (so screen will clear) */
  dmode(4); /* mid-res color graphics mode */

```

```

/* test the line algorithm by drawing in all directions, colors */

for (x=0; x<320; x+=4) plotline(160,100, x, 0, 3);
for (y=0; y<200; y+=4) plotline(160,100,319, y, 2);
for (x=319; x>=0; x-=4) plotline(160,100, x,199, 1);
for (y=199; y>=0; y-=4) plotline(160,100, 0, y, 2);

getch(); /* pause till key is pressed */
dmode(2); /* re-enter text mode */
}

*****PLOTLINE(x1,y1,x2,y2,color)
draws a line from (x1,y1) to (x2,y2) in specified color (0 to 3)
calls 'plotdot' (a fast, mid-resolution pixel-plotting routine)
*****plotline(x1,y1,x2,y2,color)
int x1,y1; /* starting point */
int x2,y2; /* ending point */
int color;
{
    /* use static variables for fastest access */
    static int lg_delta, sh_delta; /* distance of long, short axis */
    static int lg_step, sh_step; /* 0, 1 or -1 */
    static int cycle; /* decision variable */
    static int temp; /* swapping variable */

    lg_delta = x2-x1; /* get travel along X axis */
    if (lg_delta >= 0)
        lg_step = 1;
    else {
        lg_delta = -lg_delta; /* get absolute value */
        lg_step = -1; /* reverse direction */
    }

    sh_delta = y2-y1; /* get travel along Y axis */
    if (sh_delta >= 0)
        sh_step = 1;
    else {
        sh_delta = -sh_delta; /* get absolute value */
        sh_step = -1; /* reverse direction */
    }

    if (sh_delta > lg_delta) { /* if Y axis is longer, swap axes */
        cycle = sh_delta >> 1;
        temp = lg_delta; lg_delta = sh_delta; sh_delta = temp;
        temp = lg_step; lg_step = sh_step; sh_step = temp;
    }

    while (y1 != y2) { /* loop for "vertical" line */
        plotpix(x1,y1,color);
        y1 += lg_step; /* always bump line pointer */
        cycle += sh_delta; /* bump decision variable */
        if (cycle >= lg_delta) { /* past decision threshold? */
            cycle -= lg_delta; /* reset for next decision cycle */
            x1 += sh_step; /* bump column pointer */
        }
    }
    else { /* X axis is longer, so don't swap */
        cycle = lg_delta >> 1;
        while (x1 != x2) { /* loop for "horizontal" line */
            plotpix(x1,y1,color);
            x1 += lg_step;
            cycle += sh_delta;
            if (cycle >= lg_delta) {
                cycle -= lg_delta;
                y1 += sh_step;
            }
        } /* end of while */
    } /* end of else (for axis swap) */
} /* end of plotline */
}

*****DMODE(mode)
sets the display mode
mode is: 0 - bw 40x25 text 4 - color 320x200 graphics
        1 - color 40x25 text 5 - bw 320x200 graphics
        2 - bw 80x25 text 6 - bw 640x200 graphics
        3 - color 80x25 text
*****dmode(m)
int m;
{
    struct XREGS{ int ax,bx,cx,dx; } regbuf;
    regbuf.ax = m; /* AH = 0, AL = mode */
    int86(0x10,&regbuf,&regbuf);
}

```

### End Listings

# 16-BIT SOFTWARE TOOLBOX

## Recommended Software

I have been reading about the Smalltalk language with interest for years, but I couldn't quite see my way clear to applying for a job at Xerox's Palo Alto Research Center (PARC) just so I could play with it. METHODS from Digitalk is a true implementation of Smalltalk for the IBM PC (it can't be marketed with the name Smalltalk because of trademark considerations) at a reasonable price.

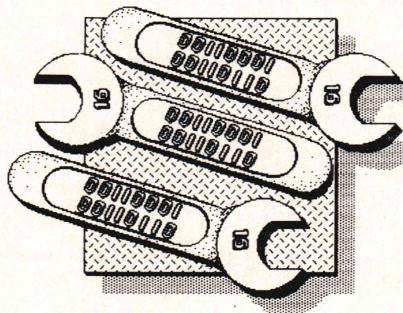
Smalltalk proper is an object-oriented programming language, but Smalltalk as people usually talk about it transcends "mere" coding to become both an environment and a philosophy for programming. The Smalltalk research and related work at Xerox's PARC have become the basis for the much ballyhooed Macintosh user interface and the other windowed, "user-friendly" operating system interfaces that are beginning to show up on microcomputers. Many of its concepts were also incorporated into the language Neon, which was developed for the Macintosh by Kriya Software in Chicago.

METHODS comes with an excellent 550-page manual that leads you gently into this brave but strange new world of object-oriented programming. METHODS' screen performance is snappy, although the implementors had to give up support for bit-mapped graphics to attain it; disk performance is OK, but I highly recommend a (large) hard disk. The arrow keys and special-function keys are used in a comfortable way to control the windows, and you can get along without a mouse quite nicely—although a mouse may be used when available.

Readers interested in learning

by Ray Duncan

more about Smalltalk should see the August 1981 special issue of *Byte* and the following books:



Goldberg, Adele. *Smalltalk-80, the Interactive Programming Environment*. Addison-Wesley, 1984.

*Smalltalk-80, Bits of History, Words of Advice*. Glenn Krasner ed. Addison-Wesley, 1983.

Goldberg, Adele, and Robson, David. *Smalltalk-80, the Language and its Implementation*. Addison-Wesley, 1983.

METHODS can be purchased for \$250 from Digitalk Inc., 5200 West Century Blvd., Los Angeles, CA 90045; (213) 645-1082.

## Additional Recommended Software

Two days after I received a copy of Microsoft C, Version 3.0, I had abandoned my Lattice C compiler forever. The new Microsoft compiler generates .EXE files that are half the size of the files generated by Lattice C—and faster besides. Microsoft C features extremely good integration with the functions of MS DOS (as you would expect, or at least hope for), including easy access to the environment block and memory management, full path support, and several variations on the exec function. I particularly like the compiler's use of SET strings in the environment to find its libraries and include files.

The Microsoft C floating point libraries use the 8087 or 80287 numeric coprocessor automatically when it is available, or in-line 8087 code can be generated. Unlike Lattice C, a true assembly-language source file can be selected as the output from the C compiler, which may be hand-optimized and then fed to the Microsoft

Macro Assembler. The documentation for the compiler, in two volumes, is far and away the best I've seen. I guess you all know where to find Microsoft, so I won't print its address and phone number here.

Pro-CED by Chris Dunford (contributor of many great programs to the public domain, including the BURN-OUT utility) is a command-line editor for PC-DOS with many powerful features including a command stack that allows you to recall and edit previously entered commands for reentry, command synonyms that allow you to abbreviate often-used commands to a few letters or symbols, the ability to chain frequently used commands together without batch files, and on-line help. Pro-CED comes with an excellent manual and is available for \$35 from Chris Dunford at P.O. Box 1072, Columbia, MD 21044; (301) 992-9371.

## Expanding the Environment

The default environment block provided by COMMAND.COM is not very big, and when you start to use programs that require several SET strings (such as the Microsoft C compiler), you may find that you exhaust the available environment space before you ever get out of the AUTOEXEC.BAT file while you are booting.

Bob Smith, author of the Tall Screen program, has discovered that the DOS 3.1 COMMAND.COM has an undocumented switch to set the size of the environment.

/E:nn sets the size of the environment area to "nn" paragraphs. Range is 10 to 62. Numbers outside that range are ignored.

The default value of the switch appears to be /E:10.

This feature is most useful when used in conjunction with the SHELL option in CONFIG.SYS. For example:

SHELL=C:\BIN\COMMAND.COM  
C:\BIN /P /E:20

### Trojan Horse Programs

Don Watkins, system operator of the CompuServe IBM PC Novice SIG, recently passed me a copy of a Trojan horse program that was uploaded to his bulletin board. This program is named DROGAN.COM and is 7,040 bytes in length. When you run it, it says "Please wait . . .", there is some disk activity, and then it displays "BYE F\_HEAD!" and exits—after formatting your disk.

Inspection of DROGAN.COM reveals that it is a "hacked" version of the IBM PC DOS FORMAT.COM program. When you download a program from a BBS without the source code, you need to be really careful—there are some twisted minds out there!

Tom Neff has compiled the following list of other reported Trojan horse programs:

DOSKNOWS.EXE—FAT killer misleadingly named the same as the harmless DOSKNOWS system-status utility. The real DOSKNOWS is 5,376 bytes long.

EGABTR—Billed as "improve your EGA display," but when run it deletes everything in sight and prints "Arf! Arf! Got you!"

FILER.EXE—labeled "Great new filing system," reportedly wiped out 20-meg hard disk.

SECRET.BAS—Formats disks.

STRIPES.EXE—Draws an American flag but copies the remote BBS configuration file to another file (STRIPES.BQS) so the uploader can call back and download all the passwords. Clever!

VDIR.COM—This is the disk killer Jerry Pournelle wrote about in *Byte* magazine.

### More on Concurrent DOS

My musings about Digital Research, GEM, and Concurrent PC DOS in recent issues of *DDJ* provoked several heated responses from readers. Evidently many people still harbor warm fuzzy feelings in their heart for Digital Research, dating back to the CP/M 1.4 days, and they suspect that DRI was somehow swindled out of its rightful place as the emperor of microcomputer operating systems. They react violently to the suggestion

that DRI might have blown the whole ball game by itself with such brilliant marketing moves as letting CP/M 2.2 stagnate for years, dawdling forever in getting a usable version of CP/M-86 into the field, and wasting valuable time and energy on hot ideas such as Dr Logo.

Well, the statements I've made about GEM and Concurrent PC DOS over the last year in this column are only my own opinions and, of course, do not reflect the opinions of the management of *DDJ*. I have been

programming on micros since the CP/M 1.4 days myself, and I certainly appreciate all the contributions Gary Kildall and DRI made with stable, relatively bugfree operating systems and compilers at the beginning of the microcomputer revolution.

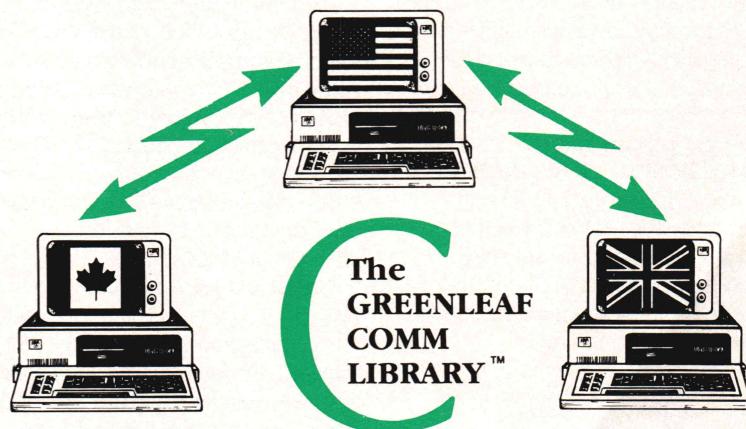
The events of the last two or three years, however, seem to point to an organization that is flailing around in the marketplace without much sense of reality or direction. Just recently we have been treated to the sight of DRI starting up massive Unix projects

## ARE YOU TRYING TO COMMUNICATE ?

C programs can communicate with the world now through the power of **The Greenleaf Comm Library**. Now from the people who brought you **The Greenleaf Functions** General Library for C, comes this rich interrupt driven, ring-buffered asynchronous communications capability.

Over 100 functions in C and assembler to facilitate communications at up to 9600 baud. Up to eight ports at a time. ASCII or XMODEM. X-On/X-Off too. Hayes compatible modems controlled here. Safe too, bet you can't exit your application with interrupts hot. Major applications around the world base their communicating applications on **The Greenleaf Comm Library**. Stop just trying and start really communicating. Get your copy of **The Greenleaf Comm Library** today. For all major C compilers, all models, all versions. For the IBM PC and just about any machine with MSDOS and an 8086. Comes with source code, extensive examples, demo programs, featuring **C-Terminal**, reference card and newsletter. No royalty. \$185

**Other Products:** **The Greenleaf Functions** General Library, over 220 functions for total control of the IBM PC, with source. \$185 for the compilers listed below. (See ordering instructions below).



**Specify compiler** when ordering: Lattice, Microsoft, Computer Innovations, Mark Williams, or DeSmet. Add \$7.00 for UPS Second Day Air (or \$5.00 for ground). Texas residents add sales tax. Mastercard, VISA, check, or P.O. In stock, shipped same day.

<input type="checkbox"/> General Libraries	\$185
<input type="checkbox"/> Comm Library	\$185
<input type="checkbox"/> CI186 Compiler	\$349
<input type="checkbox"/> Lattice C	\$395
<input type="checkbox"/> Mark Williams	\$475

For Information:  
214/446-8641



2101 HICKORY DR.  
CARROLLTON, TX 75006

PRICES ARE SUBJECT TO  
CHANGE WITHOUT NOTICE.

and then abandoning them; announcing Protected Mode Concurrent DOS-286 and then fizzling out; caving in to Apple without a battle on the GEM visual interface; dropping support for CP/M-Plus even though the 8080/Z80 operating sys-

tems are probably DRI's sole profitable product by now; and spending millions of dollars on advertising to sell only about 50,000 copies of GEM at a price that could have barely covered the costs of materials, packaging, shipping, and handling.

The Concurrent PC DOS project seems to me to be the height of foolishness and a particularly glaring ex-

ample of DRI's ability to delude itself into believing that the old days can come again. Given the close working relationship between Microsoft and IBM and the frequency with which new IBM hardware products and Microsoft MS DOS versions are released, DRI will always be playing a catch-up game. Even if Concurrent DOS performed as well as MS DOS (which it doesn't), DRI could never hope to be less than six months to a year behind in emulating the functionality of the latest revision of MS DOS—which just isn't going to be good enough for today's software market. DRI would have been much wiser to invest the money and programming talent it put into Concurrent DOS into something that the market really needed instead of into something that DRI wished the market needed.

## IBM PC Graphics

Dan Rollins, the prolific author of magazine articles on 8086 assembly language and the book *IBM PC, 8088 Macro Assembler Programming* (Macmillan, 1985), was kind enough to send us the following letter and listings:

"My interest was piqued by Tom Hogan's article 'Using Decision Variables,' published in the May 1985 issue of *DDJ*. I was disappointed in how slow the ellipsis generator executed—far slower than the BASIC CIRCLE command. The code that generates the coordinates is very fast. The bottleneck is in the pixel-plotting subroutine. I thought your readers might appreciate a fast IBM dot plotter to add to their toolboxes.

"The PLOTDOT.ASM program (Listing One, page 112) is different from similar routines in that I have made a concerted effort to cut down on clock cycles. The first thing I did was to remove any 'general code' used to determine the current screen mode, and so on. This version will work only for midresolution graphics.

"But the breakthrough came when I realized that lookup tables could be used to convert the x,y coordinate pair into a screen buffer location and pixel position. This technique eats up some extra RAM (just over 400 bytes), but silicon is cheap. The routine sacrifices storage for speed, and the trade-off is a pretty good deal. It avoids a very time-consuming multi-

# Lattice® Works

## NEW PROGRAMMER'S SCREEN EDITOR INTRODUCED

Designed specifically for programmers, the Lattice Screen Editor (LSE) is a fast and flexible, multi-window editor that is also easy to learn and use.

LSE runs under MS-DOS or PC-DOS on most popular machines with 128Kb memory. It provides standard editor functions such as block moves, pattern searches, and "cut and paste". In addition, LSE offers special features for programmers such as an error tracking mode and three assembly language input modes.

A complete installation program is included to remap any of LSE's 48 keyboard functions. Menus, prompts, help messages and default file extensions can also be customized for individual user preferences. \$125.00.

## LATTICE TOPVIEW TOOLBASKET NOW AVAILABLE

Providing more than seventy functions, the Lattice TopView Toolbasket is designed for software developers writing applications for IBM's TopView multi-tasking, multi-window environment.

The Toolbasket functions eliminate the need for extensive use of assembly language when interfacing with TopView. The Toolbasket's library includes functions to control window, cursor, pointer, and printer operations. It also provides access to TopView's cut-and-paste facilities and offers debugging services.

The Toolbasket runs on the IBM PC, XT, AT, and compatible systems with 256Kb memory. \$250.00. Binary and source code is available for \$500.00.

## LATTICE CREATES C COMPILER FOR COMMODORE AMIGA

Amiga C, produced by Lattice for the Commodore Amiga, supports the Amiga's 68000 microprocessor and offers the same high speed and extensive capabilities of the MS-DOS Lattice C compiler currently used by more than 30,000 software developers worldwide. Available from both Commodore and Lattice. \$300.00.

In addition, Lattice also offers cross compilers that allow you to develop Amiga programs on MS-DOS or UNIX systems.

Contact Lattice, to discuss your programming needs. Lattice provides C compilers and cross compilers for many environments including Tandy, Sony, Hewlett-Packard, Tandem, and IBM Mainframe. Corporate license agreements available.



**Lattice**

Phone (312) 858-7950 TWX 910-291-2190

INTERNATIONAL SALES OFFICES:

Benelux: De Vooght. Phone (32)-2-720-91-28. England: Roundhill. Phone (0672) 54675

Japan: Lifeboat Inc. Phone (03) 293-4711 France: SFDL. Phone (1) 666 1155

Circle no. 101 on reader service card.

plication, and the code is straight-line (there isn't a single conditional jump anywhere in the routine).

"A couple of points of interest in PLOTDOT.ASM. I have written it to interface with Microsoft C Small Model programs, and the start of the code shows the coding overhead required to do so. Second, I have shaved off some precious clock cycles by ignoring the standard interfacing techniques. Instead of the stack relative addressing that is normally used to access parameters, I have simply popped them off the stack into the available registers.

"Another interesting point is in how the main lookup table is generated. This is perhaps the only meaningful use of the Macro Assembler's REPT pseudo-op.

"Finally I am enclosing a Microsoft C line-drawing function (Listing Two, page 112) that illustrates the speed of the dot plotter. The 'plotline' function uses a decision variable, somewhat similar to that of Hogan's ellipsis generator. You could recode it in assembly language, but it might not be worth your effort. A disassembly shows that the Microsoft compiler generates quite efficient code, which is improved further by using static variables whenever convenient."

## More Light Reading

As I mentioned in last month's column, if you don't have a subscription to *Datamation*, you are missing half the fun in life. An example is the essay "Real Programmers Don't Use Pascal," which appeared in the July 1983 issue on page 263. It introduces itself with:

"The easiest way to tell who the Real Programmers are is by the programming language they use. Real Programmers use FORTRAN. Quiche Eaters use Pascal. Niklaus Wirth, the designer of Pascal, was once asked, 'How do you pronounce your name?' 'You can either call me by name, pronouncing it Veert, or call me by value, Worth,' he replied. One can tell immediately from this comment that Niklaus Wirth is a Quiche Eater. The only parameter passing mechanism endorsed by Real Programmers is call-by-value-return, as implemented in the IBM/370 FORTRAN G and H compilers. Real Programmers don't need abstract concepts to get their

jobs done, they are perfectly happy with a keypunch, FORTRAN IV compiler, and a beer. Real Programmers do list processing, string manipulation, accounting (if they do it at all), and artificial intelligence programs in FORTRAN. If you can't do it in FORTRAN, do it in assembly language. If it can't be done in assembly language, it isn't worth doing. . . ."

The article goes on in this vein for four pages, incidentally giving a proof (in FORTRAN, not ALGOL) that Seymour Cray, at least, is a Real Programmer.

## Blatant Plug and a Disclaimer

As many of you know, I have been working for a year now on what I hope will be the definitive book on MS DOS programming. It starts where the other books leave off. It is targeted at the experienced assembly-language programmer, and it has many detailed examples and working programs (it contains some C examples, too). Although my own company published this book briefly under the title *MS DOS Internals*, it has now been sold to Microsoft Press and will appear in your favorite bookstore under the title *Advanced MS DOS* in the spring of 1986.

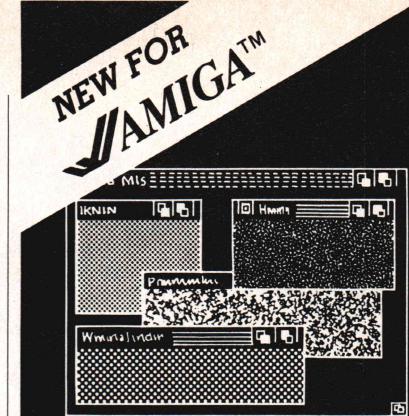
This is not only a plug but also a disclaimer. Although it is a pleasure to be associated with the fine people at Microsoft Press, let me assure you that it will not influence the way I report on Microsoft Corp. software in this column. When the company deserves praise (as with the C compiler), it will get it; and when it deserves brickbats, it'll get those too.

DDJ

## (Listings begin on page 112)

### Reader Ballot

Vote for your favorite feature/article.  
Circle Reader Service No. 7.



## MetaScope: The Debugger

MetaScope gives you everything you've always wanted in a debugger:

- Multiple Windows  
Open and close, move through memory, display data or disassembled code.
- Full Symbolic Capability  
Read symbols from files, define new ones, use anywhere.
- Powerful Expression Evaluation  
Use any standard assembler operators or number formats.
- Direct to Memory Assembler  
Enter instruction statements for direct conversion to code in memory.
- and More!  
Log file for operations and displays, breakpoint and trace execution, modify/search/find memory, etc.

MetaScope is designed to fully utilize the capabilities of the Amiga in debugging your programs. If you're programming the Amiga,™ you can't afford to be without it.

\$95 (California residents +6%).  
Visa/MasterCharge accepted.

Amiga is a trademark of Commodore-Amiga Inc.

Metadigm, Inc.

19762 MacArthur Blvd.  
Suite 300  
Irvine, CA 92715  
(714) 955-2555

Circle no. 220 on reader service card.

# ANNOUNCING! DR. DOBB'S COMPLETE TOOLBOX OF

Dr. Dobb's Journal,  
the most respected source of  
technical software information available,  
brings you this collection  
of powerful programming tools for C.

New, from M&T Publishing and  
Brady Communications . . .

## Dr. Dobb's Toolbook for C

Item #005

A comprehensive library of valuable C code.

Many of Dr. Dobb's most popular articles on C from sold-out issues are updated and reprinted in this unique reference, along with new C programming tools. The **Toolbook** contains C compilers, line editors, assemblers, text processing programs, and more! Dr. Dobb's Journal offers **The Toolbook** in a special hardbound edition for only \$29.95. You'll find:

- J.E. Hendrix's famous Small C Compiler v. 2 and A New Library for Small C (also available on disk)
- Never before published: Hendrix's new Small-Mac: An Assembler for Small C and Small Tools: Programs for Text Processing (both available on disk)
- Ron Cain's original A Small-C Compiler for the 8080's
- Plus many useful programming tools in C

Also from M&T Publishing and  
Brady Communication—  
**The Small-C  
Handbook**  
Item #006 or #006A

The **Small-C Handbook** by James Hendrix is a valuable resource of information about the Small-C Compiler. In addition to descriptions of the language and the compiler, **The Handbook** also contains the entire source listings of the compiler and its routines. A perfect companion to the Hendrix Small-C compiler offered by Dr. Dobb's on disk, **The Handbook** even tells how to use the compiler to generate new versions of itself. All this is in **The Handbook** for only \$17.95, Item #006; only \$22.95 for **The Handbook** with the MS/PC DOS **Handbook Addendum**, Item #006A.

While both **The Toolbook** and **The Handbook** provide documentation for the Small-C Compiler on disk, **The Handbook** provides more complete documentation and is available with an Addendum for MS/PC DOS versions.

# DR. DOBB'S C TOOLS ON DISK

To complement The Toolbook, Dr. Dobbs' also offers the following programs on disk. Source code is included, and except where indicated, both CP/M and MS/PC-DOS versions are available.

## Small C Compiler

Item #007

Jim Hendrix's Small-C Compiler is the most popular piece of software published in Dr. Dobbs' 10-year history. Like a home study course in compiler design, the Small-C Compiler and The Small-C Handbook provide all you need to learn how compilers are constructed, as well as teaching the C language at its most fundamental level. The Small-C Compiler is available for \$19.95 in both CP/M and MS/PC DOS versions.

Both The Toolbook and The Small-C Handbook provide documentation for the compiler; however, The Handbook provides more complete documentation for both versions, and the MS/PC DOS Small-C Handbook Addendum is recommended in addition to The Handbook for MS or PC DOS specific documentation.

## Special Packages—20% Off!

Now, for almost 20% off the individual product prices, you can order a complete set of Dr. Dobbs' C programming tools for your CP/M or MS/PC DOS system!

**CP/M C Package Only**  
\$99.95 Item #005A Ordered individually, these C tools sell for over \$120! Order the CP/M C Package now and you'll receive Dr. Dobbs' Toolbook, The Small-C Handbook, The Small-C compiler on disk, The Small Tools TextProcessor on disk, along with documentation in The Small Tools Manual, The Small-Mac Assembler on disk, and The Small-Mac Manual—all for only \$99.95!

**MS/PC DOS C Package Only**  
\$82.95 Item #005B Individual tools now sell for over \$100! Order the CP/M C Package now and you'll receive: Dr. Dobbs' Toolbook, The Small-C Handbook with the MS/PC DOS Addendum, The Small-C Compiler on disk, The Small Tools TextProcessor on disk, along with documentation in The Small Tools Manual—all for only \$82.95!

## Small-Mac: An Assembler for Small-C

Item #012A

Small-Mac is a macro assembler designed to stress simplicity, portability, adaptability, and educational value. The package features simplified macro facility, C-language expression operators, descriptive error messages, object file visibility, and an external table. Included programs are: macro assembler, linkage editor, load-and-go loader, library manager, CPU configuration utility, and dump relocatable files. This program is available with documentation in the Small Mac Manual for \$29.95. For CP/M systems only.

## Small Tools: Programs for Text Processing

Item #010A

This package consists of programs designed to perform specific functions on text files, including: editing, formatting, sorting, merging, listing, printing, searching, changing, translating, encrypting, and concatenating, copying and concatenating, replacing spaces with tabs and tabs with spaces, counting characters, words, or lines, and selecting printer fonts. Source code only is included. This program is available with documentation in the Small Tools Manual for \$29.95. Both CP/M and MS/PC DOS versions available.

Books		Dr. Dobb's Toolbook	\$29.95
Disks		Small-C Handbook	\$17.95
		Small C Handbook with MS/PC DOS Addendum	\$22.95
Books	Dr. Dobb's Toolbook		
Disks	Small-C Handbook		
	Small C Handbook with MS/PC DOS Addendum		
Books	Small-C Compiler (Toolbook or Handbook recommended for documentation)	CP/M	\$19.95
Disks	Small Tools Text Processor with Small Tools Manual	MS/PC DOS	\$29.95
Books	Small Mac Assembler with Small Mac Manual (for CP/M only)		\$99.95
Disks	SPECIAL HOLIDAY PACKAGES		\$82.95
Books	CP/M C Package		
Disks	MS/PC DOS C Package		
		Tax	Subtotal
		Shipping	%
		TOTAL	

CA residents add applicable sales tax  
Add \$1.75 per item for shipping in U.S., \$4.25  
outside U.S. Add \$8.75 shipping in U.S.  
for special C Packages.

For CP/M system disks only, please specify one of the following formats:  
 Kaypro  Apple  Zenith Z-100 DS/DD  Osborne  8" SS/SD  
Inquire about other formats.

**PAYMENT MUST ACCOMPANY YOUR ORDER.**

Check Enclosed  VISA  M/C  Amer. Exp.

Card #

Signature

Name

Address  
(Please use street address)

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Please return this coupon along with your payment to: Dr. Dobbs' Journal, 2464  
Embarcadero Way, Palo Alto, CA 94303 For credit card orders, call toll-free:  
1-800-528-6050 Ext. 4001 Refer to item numbers.

Your order will be shipped within 5 days of receipt.

3112B



# Continuing the Tradition

# DR. DOBB'S JOURNAL

## BOUND VOLUMES

Each book in this series contains a year's worth of Dr. Dobb's Journal monthly issues, reprinted in full and combined in one volume. The Bound Volumes will fill the gaps in your Dr. Dobb's collection and complete your reference library.

**Vol. 1 1976**  
The material brought together in this volume chronicles the development in 1976 of Tiny BASIC as an alternative to the "finger blistering," front-panel, machine-language programming. This is always pertinent for the bit crunching and byte saving, language design theory, home-brew computer construction and the technical history of personal computing. Topics include: Tiny BASIC, the (very) first word on CP/M, Speech Synthesis, Floating Point Routines, Timer Routines, Building an IMSA.

**Vol. 2 1977**  
These issues offer refinements of Tiny BASIC, plus then state-of-the-art utilities, the advent of PILOT for microcomputers and a great deal of material centering around the Intel 6080, including a complete operating system. Products just becoming available for reviews were the H-8, KIM-1, MITS BASIC, Poly BASIC, and NBL. Articles are about Lawrence Livermore Lab's BASIC, Alpha Micro, String Handling, Cyphers, High Speed Interaction, I/O, Tiny Pilot & Turtle Graphics, many utilities.

**Vol. 3 1978**  
This volume brings together the issues which began dealing with the 6502, with mass-market machines and languages to match. The authors began speaking more in terms of technique, rather than of specific implementations; because of this, they were able to continue laying the groundwork industry would follow. These articles relate very closely to what is generally available today. Languages covered in depth were SAM76, Pilot, Pascal, and Lisp, in addition to RAM Testers, S-100 Bus Standard, Proposal, Disassemblers, Editors.

**Vol. 4 1979**  
This volume heralds a wider interest in telecommunications, in algorithms, and in faster, more powerful utilities and languages. Innovation is still present in every page, and more attention is paid to the best ways to use the processors which have proven longevity—primarily the 8080/280, 6502, and 6800. The subject matter is invaluable both as a learning tool and as a frequent source of reference. Main subjects include: Programming Problems/Solutions, Pascal, Information Network Proposal, Floating Point Arithmetic, 8-bit to 16-bit Conversion, Pseudo-random Sequences, and Interfacing a Micro to a Mainframe.

Complete your reference library. Buy the entire set of Dr. Dobb's Journals from 1976 through 1983. Bound Volumes 1-8, for \$195.00. That's \$34.00 off the combined individual prices—a savings of almost 15%!

**Vol. 5 1980**  
Systems software reached a new level with the advent of CP/M, chronicled herein by Gary Kiddall and others (DDJ's all-CP/M issue sold out within weeks of publication). Software portability became a subject of greater import, and DDJ published Ron Cain's immediately famous Small-C compiler—reprinted here in full. Contents include: The Evolution of CP/M, a CP/M-Flavored C Interpreter, Ron Cain's C Compiler for the 8080, Further with Tiny BASIC, a Syntax-Oriented Compiler, Writing Language, CP/M to UCSD Pascal File Conversion, Run-time Library for the Small-C Compiler.

**Vol. 6 1981**  
1981 saw our first all-FORTH issue (now sold out), along with continuing coverage of CP/M, Small-C, telecommunications, and new languages. Dave Cortesi opened "Dr. Dobb's Clinic" in 1981, beginning one of the magazine's most popular features. Highlights: Information on PCNET, the Conference Tree, and The Electric Phone Book, writing your own compiler, a systems programming language, and Tiny BASIC for the 6809.

**Vol. 7 1982**  
In 1982 we introduced several significant pieces of software, including the RRD text editor and the Runic extensible compiler. Two new columns, The CP/M Exchange and The 16-Bit Software Toolbox, were launched, and we devoted special issues to FORTH and telecommunications. Resident Intern Dave Cortesi delivered his famous review of JRT Pascal and wrote the first serious technical comparison of CP/M-86 and MS-DOS. This was also the year we began looking forward to today's generation of microprocessors and operating systems, publishing software for the Motorola 68000 and the Zilog Z8000 as well as Unix code. And in December, we looked beyond, in the provocative essay, "Fifth-generation Computers."

**Vol. 8 1983**  
DDJ runs pro. Some of the most powerful, professional programmer's tools ever published in a magazine are in this volume. The second half of Jim Hendrix's Small C Compiler (continued from Vol. 7), Ed Ream's RRD screen editor. A microcomputer subset of the Defense Department's official programming language, Ada. C and Forth and 68000 software. Because the magazine increased in size in 1983, this volume is bigger and better than ever.

**YES!**  Please send me the following Volumes of **Dr. Dobb's Journal**.

Payment must accompany your order.

Please charge my:  Visa  MasterCard  American Express

I enclose  Check/money order

Card #  Expiration Date

Signature

Name  Address  (please, no P.O. Boxes)

City  State  Zip

**Mail to Dr. Dobb's Journal**, 2464 Embarcadero Way, Palo Alto, CA 94303

Allow 3-6 weeks for delivery.

3112C

Vol. 1	<input type="text"/>	x	\$26.75	=	<input type="text"/>
Vol. 2	<input type="text"/>	x	\$27.75	=	<input type="text"/>
Vol. 3	<input type="text"/>	x	\$27.75	=	<input type="text"/>
Vol. 4	<input type="text"/>	x	\$27.75	=	<input type="text"/>
Vol. 5	<input type="text"/>	x	\$27.75	=	<input type="text"/>
Vol. 6	<input type="text"/>	x	\$27.75	=	<input type="text"/>
Vol. 7	<input type="text"/>	x	\$30.75	=	<input type="text"/>
Vol. 8	<input type="text"/>	x	\$31.75	=	<input type="text"/>
All 8	<input type="text"/>	x	\$195.00	=	<input type="text"/>
				Sub-total	\$ <input type="text"/>

California residents add applicable sales tax  %

**Postage & Handling Must be Included with order.**

Please add \$2.25 per book in U.S. (\$5.25 each surface mail outside U.S. Foreign Airmail rates available on request.)

**TOTAL \$**

# PROFESSIONAL PROGRAMMER

## Software Warranties

"You have the non-exclusive right to use the enclosed program. You may not use, copy, modify, or transfer the program or documentation, or any copy, except as expressly provided in this agreement. Unauthorized reproduction, transfer, or use may be a criminal offense under federal and/or state law."

"The program is provided 'as is' without warranty of any kind. Should the program prove defective, you (and not [the company] or its dealers) assume the entire cost of all necessary servicing, repair, or correction. [The company] does not warrant, guarantee, or make any representations regarding the use of, or the results of the use of, the program in correctness, accuracy, reliability, currentness, or otherwise; and you rely on the program and results solely at your own risk."

"[The company] does warrant the disk."

"The above is the only warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose that is made by [the company]."

The above familiar prose is reproduced here without credit to (or permission of) the companies that use it. Although the whole is a composite, every sentence was taken directly from some company's shrink-wrap agreement. There's no point in singling out one or two companies when so

many use essentially the same disclaimer. It is routine in software publishing today to sell diskettes for \$400 or \$500 and inform the purchaser that there may be a piece of software on the diskette; that if there is, it may or may not do what the company's advertising says it will do; that it may or may not do it accurately, correctly, or reliably; and that it may or may not be of any use to anyone; but that in any case it is not the property of the customer, who may be tried as a criminal for any improper use of the product, proper use being defined by the manufacturer.

It's a routine that may soon be shaken—but one organization of software developers is concerned about whose hand will do the shaking.

The Software Services Association (SSA) is a California organization whose members are individuals and companies involved in software development and related fields. It's an explicitly political group that engages in lobbying for the interests of software developers. It was formed in 1982 to fight California's attempt to levy a retroactive sales tax on the work of small software companies that would extend back as far as 1974. The SSA has recently taken on the issue of software warranties, again in connection with proposed legislation.

The California legislature held hearings last summer on a bill (AB 1507) that would have required warranties on all software products. The bill was defeated, but the SSA does not view the issue as closed.

Michael Odawa of the SSA was one witness who testified at the hearings. Odawa and the SSA viewed the bill as hostile to the software industry, but Odawa has nevertheless argued in favor of voluntary adoption of warranties and against the practice of using the kind of disclaimers quoted above, contending that companies that use them "apparently feel no need to stand behind their workmanship."

Odawa argues that consumers will not continue to put up with useless warranties much longer, and he cites the legislative battle as evidence. He urges SSA members to offer good warranties and to use them as a marketing tactic. "Put a good warranty on your product," he says, "and tell the world about it. Ask your customers whether the competition stands behind its products."

The SSA publishes a newsletter that details its activities. You can get information about the organization by writing to Software Services Association, P.O. Box 6413, San Jose, CA 95150.

## Software Pricing

Another Silicon Valley organization that offers useful services and information for programmers is the Software Entrepreneurs' Forum, which meets monthly in Palo Alto. November's meeting centered on software pricing strategies.

Gregg Marshall, president of Rocky Mountain

Systems, tongue only occasionally in cheek, cited seven pricing strategies, each of which prevails over others in some market.

Several of the techniques were textbook or commonsense techniques, such as retail price equals n times cost of goods sold. Marshall set n at somewhere between 6 and 10; in other industries it is often more like 5. Other common techniques included copying the competition, pricing according to the value of the data your software juggles, and selling to the perceived value. The last idea can be particularly appealing to those in a company who see their job as manipulating the public's perception of the product. Other strategies included what Marshall called the Borland strategy of picking a magic number.

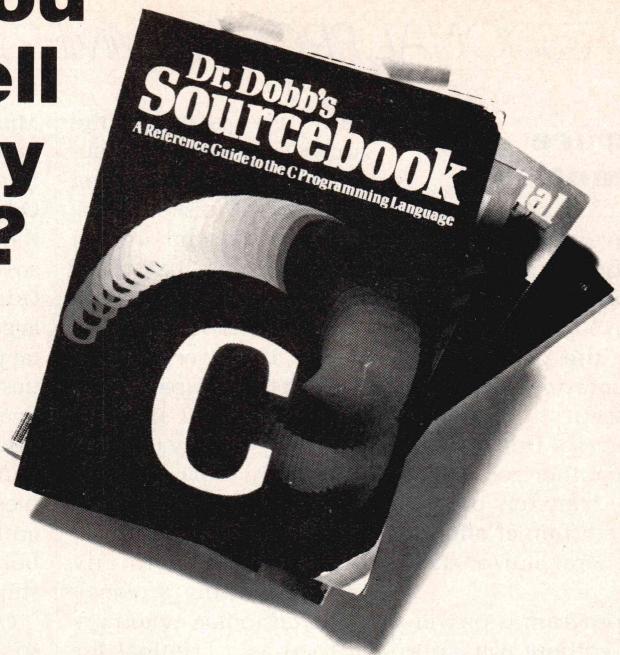
Other speakers pointed out that some price ranges create misleading impressions for consumers and that customers are willing to pay for upgrades, willing enough that there are profits to be made from upgrades. Gary Carlston, cofounder of Broderbund, emphasized the importance of involving dealers in the pricing process.

The Software Entrepreneurs' Forum has several splinter groups, such as SIGs covering technical issues for Macintosh and IBM developers, as well as marketing and engineering SIGs. The Forum also publishes a newsletter. You can find out about it by writing to Software Entrepreneurs' Forum, P.O. Box 61031, Palo Alto, CA 94306.

DDJ

# Who Says You Can't Tell A Book By Its Cover?

***Dr. Dobb's Sourcebook:  
A Reference Guide  
for the C Programming  
Language***



For years, serious programmers have relied on Dr. Dobb's Journal for the technical tools of their trade. Now, Dr. Dobb's presents the definitive programmers guide to the who, what, where, when and why of C, the leading language among software developers. This comprehensive guide to new information, products and services specific to C will be your most often-used reference!

In this valuable guide you'll find:

- An extensive directory of hardware and software services—including classes and seminars, C programming opportunities, and on-line services
- A bibliography with over 300 listings of available articles and books on C
- A comprehensive C product listing—including C compilers, graphics modules, utilities, editors and development systems, and more!
- And much more practical C programming information

At only \$7.95, no C programmer can afford to be without this unique reference.

To order by credit card, call toll free: 1-800-528-6050 ext.4001. Ask for item 004 or mail this coupon, along with payment to **Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303**

#### **PAYMENT MUST ACCOMPANY YOUR ORDER**

I enclose check/money order

Please charge my VISA M/C American Express

Card #  Exp. Date

Signature

Name

Address

City  State  Zip

Please allow 6 to 12 weeks for delivery

Please send me  copies of **Dr. Dobb's Sourcebook**

at \$7.95 each =

+ Shipping & Handling =

(Must be included with order. Please add \$1.50 per book in U.S. \$3.25 each surface mail outside U.S. Foreign airmail rates available on request.)

**TOTAL =**

## Trading Places

A flurry of post-Comdex visits to and from representatives of companies not generally thought of as software developers has left us with a curious impression: programmers and semiconductor producers are trading places.

All right, we're exaggerating. But we do so to emphasize a growing trend, one implication of which is that programmers should keep a close eye on semiconductor firms—and not just to see what new processors are coming. The design of chips is becoming more and more a software task, and the manufacturers of chips are beginning to realize that they need to be in the software business.

Take Intel. We flew up to Oregon to visit some Intel people and learned about some changes underway there.

For years, Intel has sold development systems to its customers, development systems that were, in fact, personal computers. Intel was not, however, in the commercial personal computer marketplace because its machines were built in low volumes for specific purposes. Intel was not building machines that could compete in the commercial market, nor did the company want to be in that market. Intel had decided as far back as the mid-70s that it was not a computer company and it was not in its best interest to develop a commercial computer and compete with its customers.

But the development systems Intel built, the sys-

tems it sold to its customers who were building machines around Intel's chips, were not just boxes. Intel also supplied the development software. The history of Intel's involvement in microcomputer software development is rich in intriguing might-have-beens. One of the first Intel development systems sat in the back of Gary Kildall's classroom at the Naval Postgraduate Research Institute in Monterey, California, where graduate student Gordon Eubanks wrote CBASIC as a class project. Kildall wrote some of Intel's development software, and Intel could have had CP/M if it had wanted it. But Intel didn't want a commercial product; it wasn't in the software business any more than it was in the computer business.

Intel passed up CP/M, but it did put together for each of its products a support package of the hardware and software necessary to use the product. The software, like the hardware, was designed for the development phase. In software, as in computers, Intel was just selling support tools for its products, not competing in the commercial market.

Now that's changing. Intel is coming to believe that it belongs in the commercial system-software market and that it must develop competitive products. Company employees make no mystery about the reason for the shift in strategy: it's the IBM PC/AT.

As long as you had to buy Intel's box to use Intel's chips and Intel's software was written to run on Intel's box, the firm had

a captive market for its development hardware and software. The AT has cut a link out of the chain: it provides the speed, memory, and power needed for system development, and it makes the Intel boxes unnecessary. Which leaves the software with its chain dangling. Intel development software must now compete with commercial software. That has caused some concern at Intel as well as some scrambling to get the products and the marketing of the products in line with commercial standards and practices. One Intel representative breathed a sigh of relief that *DDJ* had not reviewed the Intel C compiler in its August 1985 review but expressed interest in having the next version included in the next review.

The other side of this coin is that the design of chips (and of computers) is becoming more and more a software task. Two examples of this are silicon compilers and the software simulation of semiconductor devices.

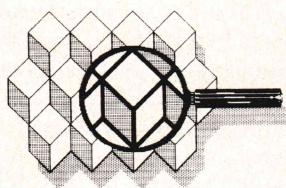
We went to hear David Johannsen speak during Comdex. Johannsen is generally credited with inventing the concept of silicon compilation as a graduate student at Cal Tech. He subsequently cemented his right to use the term by starting a company called Silicon Compilers, which develops silicon compilers (as one might naively guess) and, more recently, silicon compiler compilers. Johannsen's approach to computer-aided design most blatantly underscores the parallels between semiconductor design and software design.

Silicon compilation helps in the design and design testing of a chip just as a software compiler helps in the design of a program. The user works with low-level primitives such as ALUs, RAMs, and PLAs, building these into blocks and modules and perhaps into chips that serve as board-level primitives for a higher level of design. The notion of multiple levels is central to the silicon compiler; the user can design at different levels, with the system simulating devices down only to the level at which design is taking place.

Silicon compilation offers some capabilities that software compilers don't and perhaps should. Users of a silicon compiler can specify designs in language format, just as when writing code (or constructing truth tables, logic equations, microcode; Johannsen classifies all these approaches as language formats). But they can also work at a graphic, more intuitive level, specifying via schematics. And for conventional components, the user has the option of simply filling in a displayed form with on line help—programming by menus.

Silicon Compilers' silicon compilers provide some other softwarelike capabilities, including a rough equivalent of syntax checking in a logical design rule checker (LDR). The LDR checks for timing inconsistencies and type inconsistencies (clock expected, ground wired) and flags certain "questionable practices" (mostly nonportable stuff).

DDJ



This issue focuses on Pascal, Ada, and Modula-2 as structured languages. There has for some time been a language available in the 6502 world that embodies such structured features as syntactically meaningful indentation: Promal.

Systems Management Associates announced that its Promal 2.0 integrated programming system will support the IBM PC and compatibles by March 1986. Significant performance and capacity enhancements are expected. The product consists of a high-performance, compiled structured language; full-screen editor; operating system executive (except IBM version); and library subroutines. It currently supports the Apple IIc or IIe (with 128K and 80-column card) and the Commodore 64/128.

The Commodore and Apple versions are priced between \$50 and \$100. The IBM PC line version costs \$99.95 plus \$5 for shipping and handling.

Control-C Software has announced a software product that permits specific IBM PC applications such as Lotus 1-2-3, MultiMate, WordStar, and SideKick to work on personal computers that are not IBM-compatible. Known as Softclone, this software requires no changes to the original release disk, so any copy protection used remains in force.

At the heart of Softclone is a smart loader program that loads the IBM PC application and surrounds it with a fake IBM PC software/hardware environment. Whenever an application accesses IBM-specific memory or hardware, Softclone intervenes, accessing the nonclone host memory, hardware, and operating system instead. Under a licensing agreement, the per-CPU royalty is \$10.

Advanced Trace86 from Morgan Computing provides assembly-language programmers with a debugging environment for the IBM PC or IBM PC/AT. In Trace mode, it displays a full screen (16-22 lines) of disassembled code. An inverse video bar marks the instruction that is currently executing. Windows show a constant display of registers and flags, the current stack, and up to six lines of memory (which you can set to track program references). An optional window displays the contents of 8087 registers. It is possible to single step, trace at normal trace speed (about 30 instructions per second) or quick trace speed (about 1/3000 of native), use a keyboard interrupt to start tracing a running program, or set a breakpoint in the code. Advanced Trace86 can also step backward up to 20 instructions to get a replay of the action.

Real-Time Computer Science Corporation has announced several new products. The PL/M Connection is an interface library of more than 150 functions and utilities that provides a direct connection between an IBM PC

and Intel's PL/M 86 compiler. PL/M Connection comes on a 360K IBM PC DOS formatted disk, complete with source code in PL/M and assembly, demonstration programs (also with source code), and a 200-page manual. The license fee is \$295 per user system.

Version 3.0 of iSIM85 software allows developers to run Intel 8-bit compilers, assemblers, and utilities on IBM PC/XT/ATs or any MS DOS-based computer. It supports Intel development software for the 8085, 8080, 8089, and 8048/8051 processor lines.

Intel and Advanced Computer Techniques have agreed to develop an 80286-based Ada compiler. Aimed at military software design applications, the Ada-286 compiler will support two target environments: the 80286 microprocessor chip and the iRMX286 operating system. Both environments will support the M80287 numeric data processor.

The Advanced Logic Research System 286 is based on the advanced 80286-8 16-bit microprocessor with a system clock rate of 8 MHz. It is designed to be IBM PC/AT bus compatible with full attention to the BIOS ROMs. The system supports IBM PC DOS 3.0 and 3.1 and Xenix operating systems.

A 5-inch board that fits into one IBM "short" slot, the PC/Short Memory from Emulex expands the IBM PC or compatibles to a full memory capacity of 640K. It offers starting addresses of 128K, 256K, 384K, and 512K so the board can work on a variety of personal computers with different RAM capaci-

ties on their motherboards. The PC/Short Memory also supports byte parity to protect data in the event of a failure.

Micro Computer Technologies has introduced what it calls the next generation of expansion boards for the IBM PC/XT. The Modular Expansion Series includes an externally mounted visual/audible indicator for keyboard, keylock positions, fast/slow speed, programmable key function, and an added four feet of keyboard extension cable. No IBM expansion slot is required.

The power to develop transportable applications on a range of hardware, including the IBM PC AT, IBM PC XT, and Unix machines, is available through Progress from Data Language. Progress is an application development system combining a fourth-generation application language with a database management system. Advanced facilities provide a single productive environment for building transaction-oriented applications without conventional programming. Its multiuser, environment supports simultaneous database access and update while maintaining database integrity through systems failures.

The Cauzin Softstrip System is for entering, storing, and distributing data. It is an alternative to diskettes and telecommunications. The Softstrip data strips are decoded and entered into a personal computer via an optoelectronic scanning device that plugs into IBM, Apple II series, and Macintosh personal computers. Each data strip can

hold up to 5,500 bytes and can be linked for lengthy programs, financial information, or databases.

Advanced Digital Corporation's PC-Slave II is a high-speed (8 MHz), 16-bit single-board processor that contains two Intel 80188 CPUs, two 512K memory chips, 2K of dual port networking, and a priority interrupt controller. It achieves both hardware and software compatibility with the IBM PC and compatibles by providing functional identity I/O addresses, video display, keyboard interface, and operating system software. Single-user systems can be expanded to multiuser systems of 2-32 individual workstations.

Sumitronics has released the General Purpose Cross Assembler, XASS-V. In addition to supporting programming for all common microprocessors, the XASS-V also supports custom CPUs. Definition files are generated by the option program of the Definition Processor (XGEN-V). It is executable in the 32-bit native mode of VAX-II series and micro VAX-I/II.

Version 1.5 of Koch Software's PC Sweep disk/file management program features an unerase capability that supports fixed and removable hard disks. PC Sweep is an enhanced MS DOS version of the Sweep CP/M software program, and runs on the IBM PC/XT/AT. The product allows users to search for files and determine memory usage of each file, view or page through document files, and access the main menu of help screens via one-key commands.

WATFOR-77, the latest member of the WATFOR family of FORTRAN compilers, has been released by WATCOM Products for the

IBM PC DOS operating system. By eliminating the need to produce object files and the linking process, WATFOR-77 can be used to develop and debug programs. The PC version is available for a one-time license fee of \$295. A site can be licensed for \$1,200 (up to 20 computers) or for \$3,000 (more than 20 computers) annually.

Micro Data Base Systems has introduced Guru, an artificial-intelligence software package developed for businesses operating in an IBM PC environment. Guru integrates expert systems capabilities and a natural-language interface with business tools such as spreadsheets, database managers, telecommunications, and text processors.

Multiple ports on the Microfazer II from Quadram provide parallel/parallel, parallel/serial, serial/serial, and serial/parallel modes in a single unit. It has an 8K memory that can be expanded in increments of up to 2 megabytes. Microfazer II buffers print data without using up any of the computer's memory. It then takes over the printing task, freeing the computer.

The CLASIX MultiDrive Director from Reference Technology is designed to increase the number of CD-ROM drives attached to a personal computer. The Director connects up to eight DataDrive Series 500 optical disk drives to a single IBM PC/XT/AT or compatible.

MicroMotion MasterFORTH 1.2 for the IBM PC line now supports the 8087/80287 math coprocessors. The 8087 extension includes a macro assembler with local labels supporting all precisions, opcodes, and synchronization. The

floating-point package includes transcendental and high-level functions as well as formatted input and output routines. MasterFORTH 1.2 includes a full file interface to MS DOS 2.1-3.1. It is also available for the Macintosh, the Apple II line, the Commodore 64, and CP/M machines.

### Protecting Data

What happens to all that information in the event of a power failure? Elgar has released Failsafe, a microcomputer power-protection device. When utility power fails, Failsafe is activated. After power is restored, it returns the program to where it left off and completes unfinished tasks.

Fastback (Version 5.0) is a hard disk backup software utility from Fifth Generation Systems that backs up a 10-megabyte hard disk on standard 5 1/4-inch floppies or on the IBM PC/AT. It works with PC DOS or MS DOS, Versions 2.0 or later.

Emerald Systems' Archival Storage Protector supports Novell local area networks, including Netware special files. The utility is available with Emerald 1/4-inch tape backup subsystems for both stand-alone and networked IBM-compatible microcomputers. Volumes range from 30 to 236 megabytes of formatted capacity.

Everex has added two memory expansion boards for the IBM PC/AT and compatibles. The RAM 2500AT holds 2 1/2 megabytes of memory (using 64K RAM chips), and the RAM 3000 AT holds 3 megabytes and can use 64K or 256K chips.

Kustom Electronics has introduced the Sunflower MS 10 IR, an internal 10-megabyte removable hard disk kit designed for the IBM PC/XT/AT and AT&T

6300. It is suitable for companies and government agencies handling sensitive or classified data.

### Communications

Network Revelation from Cosmos is a transaction-oriented applications environment for microcomputer networks. It uses two filing systems: ROS and Link. ROS is designed for single-user and unshared network files. Link arranges data into 1K frames and allows network locking routines to be used.

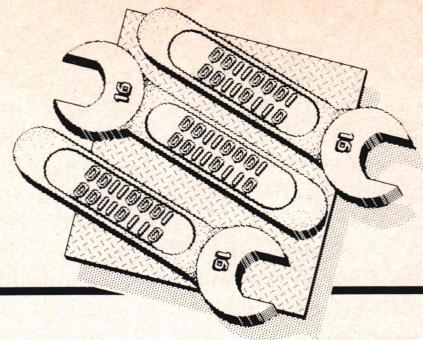
The 1200PA, a 212A-compatible modem from Radical-Vadic is designed to solve application problems encountered in dial-up modems. A full dialing keyboard permits control, option setting, dialing, and running diagnostics from the front panel, eliminating pencil switches or dependence upon terminal protocols.

Torus' icon-based networking software system, Tapestry, runs unmodified on IBM's Token Ring networking hardware. The product integrates the functions of file and record management, electronic mail, an independent library of single-user and multiuser applications software, telephone management, shared printers and modems, and other tools.

The Zoom Telephonics Zoom/Modem PC 1200 for the IBM PC/XT/AT and compatibles features call-progress tone detection, real-time clock/calendar, auto-answer touch-tone password security, audio input port, RAM buffer for background electronic messaging, support of four COM ports, and a high-speed 16450 UART for compatibility with the IBM PC/AT and AT clones.

The Software Link's

**Z80 CP/M USERS TAKE HEART!  
HERE'S ALL YOU NEED TO WRITE  
YOUR OWN PROGRAMS FOR ONLY:  
\$25.00!**



# DR. DOBB'S Z80 TOOLBOOK

By David E. Cortesi

Do you use CP/M? Do you feel as if the only part of the computer industry that has not abandoned you is your own Z80 computer? It keeps on working, but when you need programs for it, you have to write

them yourself. When you do, you quickly find that while Pascal or Basic is okay for some things, there is often no substitute for the speed, small size, and flexibility of an assembly language program.

**DR. DOBB'S Z80 TOOLBOOK** puts the power of assembly language in the hands of anyone who's done a little programming. You'll find:

- **A method of designing programs and coding them in assembly language**—and—a demonstration of the method in the construction of several complete, useful programs.
- **A complete, integrated toolkit of subroutines** for arithmetic, for string-handling, and for total control of the CP/M file system. They bring the ease and power of a compiler's runtime library to your assembly language work, without a compiler's size and sluggish code.

Best of all, every line of the toolkit's source code is there for you to read, and every module's operation

is explained with the clarity and good humor for which Dave Cortesi's writing is known.

## Order the Z80 Software on Disk! Save Yourself the Frustration of File Entry

All the software in **DR. DOBB'S Z80 TOOLBOOK**—the programs plus the entire toolkit, both as source code and as object modules for both CP/M 2.2 and CP/M Plus—is yours on disk. (A Z80 microprocessor and a Digital RMAC assembler or equivalent are required.)

Receive **DR. DOBB'S Z80 TOOLBOOK**, along with the software on disk, together for only \$40.

To order, return this form with your payment, plus \$1.75 for shipping in the U.S., \$3.75 outside the U.S., to: M&T Publishing, 2464 Embarcadero Way, Palo Alto, CA 94303.

Or, for faster service on credit card orders, CALL TOLL FREE 1-800-528-6050, ext. 4001. Refer to item #022 for the **Z80 Toolkit** and item #022A for the **Z80 Toolkit** with the disks. Please specify one of the disk formats offered below.

Send me  copies of **DR. DOBB'S Z80 TOOLBOOK**:

\$25 ea

Send me  sets of **DR. DOBB'S Z80 TOOLBOOK**  
along with the software on disk  
for \$40 per set:

\$40 ea

Subtotal

(CA residents add applicable  %  
sales tax.)

Shipping

TOTAL

For the Z-80 software on disk, please specify one of the following formats:

8" SS/SD  Osborne

Apple  Kaypro

Check enclosed  Charge my  Amer. Exp.  
 VISA  M/C

CARD #  EXP. DATE

SIGNATURE

NAME

ADDRESS

CITY  STATE  ZIP

Your order will be shipped within 5 days of receipt.

## OF INTEREST

(Continued from page 125)

MultiLink Advanced and LANLink provide a software-driven solution to multiuser and shared-resource systems for IBM PCs and compatibles. MultiLink Advanced allows you to use up to eight dumb terminals with one microcomputer using a floppy and the standard RS-232 port. It can be combined with LANLink to create a companywide system with as many as 72 workstations operating concurrently.

### Reference Map

Systems Management, 3325 Executive Dr., Raleigh, NC 27609; (919) 878-3600, (800) 762-7874. Reader Service Number 16. Control-C Software, 6441 S.W. Canyon Ct., Portland, OR 97221; (503) 292-8842. Reader Service Number 17. Morgan Computing, P.O. Box 112730, Carrollton, TX 75011; (214) 245-4763. Reader Service Number 18. Real-Time Computer Sci-

ence Corp., 1390 Flynn Rd., Camarillo, CA 93010; (805) 987-9781. Reader Service Number 19.

Intel, 5000 W. Williams Field Rd., Chandler, AZ 85224; (602) 961-8420. Reader Service Number 20.

Advanced Logic Research, 2991 E. White Star Ave., Anaheim, CA 92806; (714) 666-2951. Reader Service Number 21.

Emulex, 3545 Harbor Blvd., P.O. Box 6725, Costa Mesa, CA 92626; (800) 854-7112, (714) 662-5600 in Calif. Reader Service Number 22. Micro Computer Technologies, 1745 21st St., Santa Monica, CA 90404; (213) 829-3641. Reader Service Number 23.

Data Language, 47 Manning Rd, Billerica, MA 01821; (617) 663-5000. Reader Service Number 24.

Cauzin Systems, 835 South Main St., Waterbury, CT 96706; (203) 573-0150. Reader Service Number 25.

Advanced Digital Corp., 5432 Production Dr., Huntington Beach, CA 92649; (714) 891-4004, (800) 251-1801. Reader Service Number 26.

Sumitronics, 580 N. Pastoria Ave., Sunnyvale, CA 94086; (408) 737-7683. Reader Service Number 27.

Koch Software, 11 W. College Dr., Bldg. G, Arlington Heights, IL 60004; (312) 398-5440. Reader Service Number 28.

WATCOM Products, 415 Phillip St., Waterloo, ONT N2L 3X2; (519) 886-3700. Reader Service Number 29.

Micro Data Base Systems, P.O. Box 248, Lafayette, IN 47902; (317) 447-1122. Reader Service Number 30.

Quadram Corp., One Quad Way, Norcross, GA 30093-2918; (404) 923-6666. Reader Service Number 31.

Reference Technology, 1832 N. 55th St., Boulder, CO 80301; (303) 449-4157. Reader Service Number 32. MicroMotion, 8726 Sepulveda Blvd., #A171, Los Angeles, CA 90045; (213) 821-4340. Reader Service Number 33.

Elgar, 9250 Brown Deer Rd., San Diego, CA 92121; (619) 450-0085. Reader Service Number 34.

Fifth Generation Systems, 909 Electric Ave., Ste 202, Seal Beach, CA 90740; (800)

225-2775, (213) 439-2191.

Reader Service Number 35.

Emerald Systems, 4757 Morena Blvd., San Diego, CA 92117; (619) 270-1994. Reader Service Number 36.

Everex, 47777 Warm Springs Blvd., Fremont, CA 94539; (415) 498-1111. Reader Service Number 37.

Kustom Electronics, 8320 Nieman Rd., Lenexa, KS 66214; (800) 255-6311. Reader Service Number 38.

Cosmos, 19530 Pacific Highway South, Seattle, WA 98188; (206) 824-9942. Reader Service Number 39.

Racal-Vadic, 1525 McCarthy Blvd., Milpitas, CA 95035; (408) 946-2227. Reader Service Number 40.

Torus, 411 Seaport Ct., Ste. 105, Redwood City, CA 94063; (415) 363-2418. Reader Service Number 41.

Zoom Telephonics, 207 South St., Boston, MA 02111; (617) 423-1072. Reader Service Number 42.

The Software Link, 8601 Dunwoody Pl. NE, Ste. 632, Atlanta, GA 30338; (404) 998-0700. Reader Service Number 43.

—Wendelin Colby  
DDJ

## ATTENTION TURBO PASCAL PROGRAMMERS

Excellent product... best screen manager I've ever seen in any language.

Don A. Williams  
Finance & Administration  
Honeywell

**TURBO PROFESSIONAL**



The word professional most certainly describes the capabilities this subroutine library provides for implementing interrupt service and pop-up routines.

Computer Language Magazine

TURBO PROFESSIONAL's easy to use routines let you... write safe, "sidekickable" routines that can use DOS

- execute DOS commands from Turbo
- create super-fast windowed displays
- service interrupts without assembler
- make your own keyboard enhancers
- allocate memory from DOS
- print concurrently with DOS 3.x

Complete with 109+ routines, manual, source code to Super Macs keyboard enhancer, and many example programs.

Only \$49.95 + \$5.00 shipping & handling

Visa, MasterCard ok.

**Order now!**

**(206) 367-0650**



Sunny Hill Software  
13732 Midvale North Suite 206  
Seattle, Washington 98133

Get the tools that have no competition.

Requires Turbo for compatibles. Turbo Pascal & Sidekick Trademark Borland Int'l.

Circle no. 172 on reader service card.

# DeSmet

## C

**8086/8088  
Development  
Package** **\$109**

### FULL DEVELOPMENT PACKAGE

- Full K&R C Compiler
- Assembler, Linker & Librarian
- Full-Screen Editor
- Execution Profiler
- Complete **STDIO** Library (>120 Func)

### Automatic DOS 1.X/2.X SUPPORT

### BOTH 8087 AND S/W FLOATING POINT OVERLAYS

### OUTSTANDING PERFORMANCE

- First and Second in AUG '83 BYTE benchmarks

### SYMBOLIC DEBUGGER

**\$50**

- Examine & change variables by name using C expressions
- Flip between debug and display screen
- Display C source during execution
- Set multiple breakpoints by function or line number

### DOS LINK SUPPORT

**\$35**

- Uses DOS .OBJ Format
- LINKs with DOS ASM
- Uses Lattice® naming conventions

Check:  Dev. Pkg (109)

Debugger (50)

DOS Link Supt (35)

SHIP TO: \_\_\_\_\_

ZIP \_\_\_\_\_

**CWARE**  
CORPORATION

P.O. BOX C  
Sunnyvale, CA 94087  
(408) 720-9696

All orders shipped UPS surface on IBM format disks. Shipping included in price. California residents add sales tax. Canada shipping add \$5, elsewhere add \$15. Checks must be on US Bank and in US Dollars. Call 9 a.m. - 1 p.m. to CHARGE by VISA/MC/AMEX.

Street Address: 505 W. Olive, #767. (94086)

## ADVERTISER INDEX

Reader Service	Page	Reader Service	Page		
No.	Advertiser	No.	Advertiser		
158	AMPRO Computers, Inc.	78	105	Microprocessors Unlimited	90
221	Alcyon Corp.	39	190	Mitek	55
249	Apple Computer	59	*	Mix Software	65
121	Arity Corp.	15	238	Modula Corp.	80
216	Atron	50	128	Morgan Computing	88
159	Blaise Computing	19	191	Northwest Computer Algorithms	105
161	Borland International	C4	124	Optotech	1
181	C Users Group	88	192	Overland data Inc.	82
*	C Ware	188	200	Pecan Software Systems, Inc.	21
226	Cauzin Systems, Inc.	4-5	239	Perfect Market, Inc.	106
209	Certified Software Corp.	98	76	Personal Tex, Inc.	51
178	Chalcedony	54	139	Phoenix Computer Products	57
81	Cogitate, Inc.	90	91	Phoenix Computer Products	C-3
237	CompuServe	13	193	Plu Perfect Systems	78
122	CompuView	33	169	Poor Person Software	81
225	Computel Publishing Society	109	229	Port A Soft	84
232	Computer Faire, Inc.	77	*	Precise Electronics	83
96	Computer Innovations	71	140	Productivity Products Int'l	46
82	Creative Programming	37	143	Programmer's Shop	24,25
86	Dalsoft Systems	88	141	Programmer's Shop	29
203	Datalight	103	107	Quilt Computing	90
87	Digital Research Computers	31	145	Rational Systems	66
235	Drafrronics	67	213	68 Micros	98
179	Earth Computers	57	*	SAS Institute Inc.	11
89	Ecosoft, Inc.	23	78	SLR Systems	64
90	Edward K. Ream	103	114	Seidl Computer Engineering	81
138	Essential Software	48	85	Semi Disk Systems	111
165	Everest Solutions	C-2	241	Sharpe Systems Corp.	110
180	Executive Systems	69	219	Simon & Schuster Computer	
93	FairCom	58	202	Books	17
94	Fox Software	44		Simon & Schuster Computer	
*	Gimple Software	38		Books	93
*	Gimple Software	52		Soft Advances	86
97	Greenleaf Software	115	*	Soft Focus	93
98	Guidance Software	99	88	Softaid	107
132	Harvard Softworks	68	153	Solution Systems	49
134	HiSoft	84	148	Solution Systems	49
*	House ad (DDJ Bound Volume)	120	152	Solution Systems	53
*	House ad (DDJ C-Products)	118-119	155	Solution Systems	53
*	House ad (DDJ Sourcebook)	122	147	Solution Systems	45
*	House ad (DDJ Z80 Toolbook)	126	240	Southern Pacific Computer Prod.	14
*	House ad (DDJ Back Issues)	101	236	Springer-Verlag Publishers	72
*	House ad (DDJ Allen Holub)	79	164	Spruce Technologies	97
*	House ad (DDJ Mac Reprint)	92	228	Summit Software Technology	35
*	House ad (DDJ Subscription)	100	172	Sunny Hill Software	127
194	InfoPro Systems (Trade)	108	104	System Management Assoc.	59
*	Integral Quality	85	174	TLM Systems	49
101	Lattice, Inc.	116	175	TLM Systems	51
135	Lugaru Software Ltd.	47	173	TLM Systems	53
218	MacMemory Inc.	110	245	Tech PC	41
222	Manx Software	73	231	Terra Base Software	109
223	Manx Software	74	247	The COBOL Shop	97
109	Manx Software	22	243	The Norton Utilities	51
108	Manx Software	7	234	Think Technology Inc.	40
102	Mark Williams	2	207	Turbo Power Software	91
189	Martian Technologies	80	77	UniPress Software	43
167	Martin Scot Development	89	157	Vermont Creative Software	87
220	Metadigm, Inc.	117	112	Wendin, Inc.	9
110	Micro Interface Corp.	82	116	Wizard Systems	107
*	MicroMethods Inc.	95	244	Workman & Associates	108
136	Microcomputer Systems Cnslts	89	*	Worldwide Access Inc.	75
*	Micromint, Inc.	70			

\*This advertiser prefers to be contacted directly: see ad for phone number.

### Advertising Sales Offices

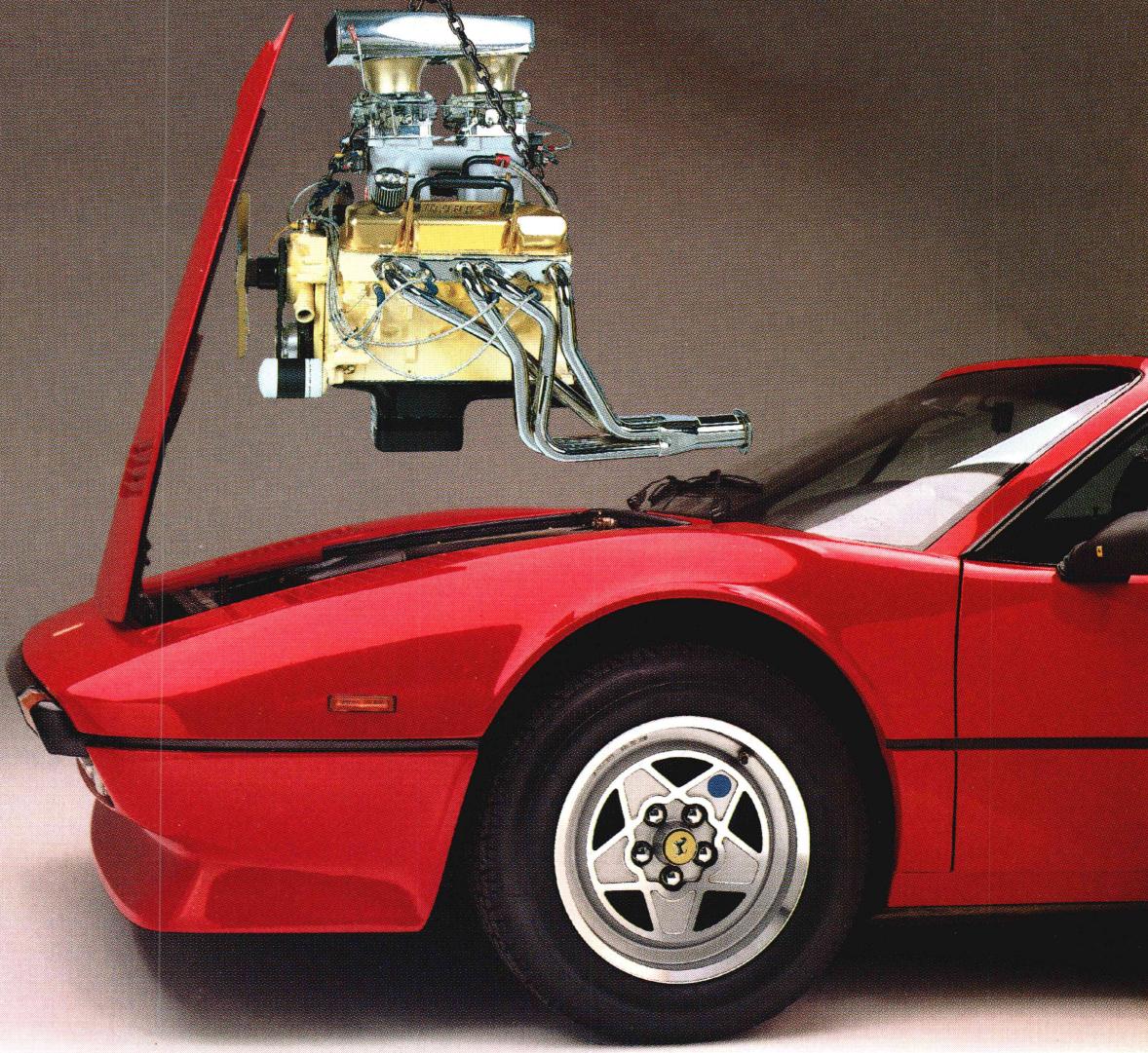
**East Coast**  
Walter Andrzejewski (617) 567-8361

**Northern California/Northwest**  
Lisa Boudreau (415) 424-0600

**Midwest**  
Michele Beaty (317) 875-8093

**Southern California/AZ/NM**  
Beth Dudas (714) 643-9439

**Advertising Director**  
Shawn Horst (415) 424-0600



## AT™ Pfantasies for your PC or XT.™

Want better speed and memory on your PC or XT without buying an AT?

You've got it!

Phoenix's new Pfaster™286 co-processor board turns your PC or XT into a high-speed engine 60 percent faster than an AT. Three times faster than an XT. It even supports PCs with third-party hard disks. But that's only the beginning.

You can handle spreadsheets and programs you never thought possible. Set up RAM disks in both 8088 and 80286 memory for linkage editor overlays or super-high-speed disk caching. All with Pfaster286's 1mb of standard RAM, expandable to 2mb, and dual-mode design.

You can develop 8086/186/286 software on your XT faster. Execute 95 percent of the application packages that run on the AT, excluding those that require fancy I/O capabilities your PC or XT hardware just isn't designed to handle. Queue multi-copy, multi-format print jobs for spooling. Or, switch to native 8088 mode to handle



hardware-dependent programs and back again without rebooting. All with Pfaster286's compatible ROM software. And, Pfaster286 does the job unintrusively! No motherboard to exchange. No wires to solder. No chips to pull. Just plug it into a standard card slot, and type the magic word, "PFAST."

If you really didn't want an AT in the first place, just what it could do for you, call or write: Phoenix Computer Products Corp., 320 Norwood Park South, Norwood, MA 02062; (800) 344-7200. In Massachusetts, 617-762-5030.

Programmers' Pfantasies™  
by

*Phoenix*

XT and AT are trademarks of International Business Machines Corporation. Pfaster286 and Programmers' Pfantasies are trademarks of Phoenix Computer Products Corporation. For the Ferrari aficionado: yes, we know this is a rear engine car. We are showing the addition of a second engine to symbolize how Pfaster can be added to your PC or XT to increase performance.

SAVE OVER 30% ON OUR GIFT PACKS!  
60-DAY MONEY-BACK GUARANTEE

# How Borland's Three New Holiday Packs Will Fill Your Stocking Without Emptying Your Piggybank.

Three special packs with dazzling discounts that will help get you into a Holiday mood. You can get some of Turbo, most of Turbo, or all of Turbo—including the two newest members of the Turbo family, Turbo GameWorks™ and Turbo Editor Toolbox™. You also get our unmatched 60-day money-back guarantee, quality products that aren't copy-protected.

## TURBO NEW PACK \$95.00.

You get the two exciting new members of the Turbo Pascal family,

- TURBO GAMEWORKS, Chess, Bridge, and Go-Moku, complete with source code and a 200-page manual.
- TURBO EDITOR TOOLBOX, all the building blocks to make your own editors and word processors, complete with source code and a 200-page manual.

## TURBO HOLIDAY PACK \$125.00.

You get all three of the Turbo family classics for only \$125.00 (about a 30% discount). Turbo Pascal 3.0 and Turbo Tutor and Turbo DataBase Toolbox—all for just \$125.00.

- TURBO PASCAL combines the fastest Pascal compiler with an integrated development environment.
- TURBO TUTOR teaches you step-by-step how to use Turbo Pascal with commented source code for all program examples on diskette.
- TURBO DATABASE TOOLBOX offers three problem-solving modules for your Turbo Pascal programs: Turbo Access, Turbo Sort, and GINST, which generates a ready-to-run installation program that lets you forget about adapting your software to specific terminals.

## TURBO HOLIDAY JUMBO PACK \$245.00.

This is it—the whole thing, the entire Turbo family including its two newest members. You get:

- Turbo Pascal
- Turbo Tutor
- Turbo GameWorks
- Turbo Graphix Toolbox
- Turbo DataBase Toolbox
- Turbo Editor Toolbox

and you pay only \$245.00 for all six! Which means that you're getting everything at only about \$40 a piece. Quite a holiday deal. (And if you already own one or several members of the Turbo family, be creative—nothing can stop you from buying the Jumbo Pack, picking out the ones you already have and giving the rest as holiday gifts to family and friends. At these prices you can afford to give to others and to yourself.) Speaking of Holidays, this offer lasts until March 31, 1986. (At Borland, we like to make the Holidays last.)



4585 SCOTTS VALLEY DRIVE, SCOTTS VALLEY,  
CA 95066 PHONE (408) 438-5400 TELEX 172373

Copyright 1985 Borland International! BI-1017B

Turbo Pascal and Turbo Tutor are registered trademarks and Turbo DataBase Toolbox, Turbo Graphix Toolbox, Turbo Editor Toolbox, Turbo GameWorks, and MicroStar are trademarks of Borland International, Inc. WordStar is a trademark of MicroPro International Corp. Multi-Mate is a trademark of Multimedia International Corp. Microsoft is a registered trademark and Word is a trademark of Microsoft Corp. WordPerfect is a trademark of Satellite Software International.



NEW!

## TURBO GAMEWORKS

\$69.95.

Our new Turbo GameWorks offers games you can play and replay without Turbo Pascal or revise and rewrite with Turbo

Pascal 3.0. We give you the source code, the manual, the diskettes and the competitive edge. Chess, Bridge and Go-Moku. State-of-the-art games that let you be player, referee, and rules committee all at once because you have the Turbo Pascal source code. Learn exactly how the games are made—so you can go off and make your own. And Turbo GameWorks is the only quality game you can buy that is not copy-protected. Sold separately, only \$69.95. (Just \$47.50 if you buy the Turbo New Pack.)

## NEW! TURBO EDITOR TOOLBOX

\$69.95.

Build your own word processor—for only \$69.95!

You get ready-to-compile source code, a full-featured WordStar™-like word processor, and a 200-page manual that tells you how to integrate the editor procedures and functions into your programs. With

Turbo Editor Toolbox, you can have the best of all word processors. You can make WordStar behave like Multi-Mate. Support windows just like Microsoft's Word. And do it as fast as WordPerfect does it. Incorporate your new "hybrids" into your programs to achieve incredible control and power. Sold separately, only \$69.95. (If you buy the Turbo New Pack, the price drops to just \$47.50.)

NOT COPY-PROTECTED

## Holiday Gift Packs, Turbo GameWorks™ & Turbo Editor Toolbox™

Available at better dealers nationwide. Call (800) 556-2283 for the dealer nearest you. To order by  
Credit Card call (800) 255-8008, CA (800) 742-1133.

Make checks payable to:  
Borland International.

The holiday packs include an upgrade coupon for both options so you get BCD and 8087 support for \$39.95 (regularly \$55.00).

Carefully describe your computer system!  
Mine is:  8-bit  16-bit  
I use:  PC-DOS  MS-DOS  
       CP/M-80  CP/M-86

My computer's name and model is: \_\_\_\_\_

The disk size I use is:  3 1/2"  5 1/4"  8"

Name: \_\_\_\_\_

Shipping Address: \_\_\_\_\_

City: \_\_\_\_\_ Zip: \_\_\_\_\_

State: \_\_\_\_\_ Telephone: \_\_\_\_\_

\*Gift Pack  
Offers Last  
Until March 31,  
1986

Quantity

*Turbo Holiday Jumbo Pack	\$245.00
*Turbo Holiday Pack	\$125.00
*Turbo New Pack	\$69.95
Pascal	\$109.90
Pascal w/8087	\$109.90
Pascal w/BCD	\$124.95
Pascal w/8087 and BCD	\$124.95
Turbo DataBase	\$54.95
Turbo Graphix	\$34.95
Turbo Tutor	\$69.95
Turbo Editor	\$69.95
Turbo GameWorks	\$69.95

These prices include shipping to all U.S. cities. All foreign orders add \$10 per product ordered.

Amount: (CA add 6% tax) \_\_\_\_\_

Payment: VISA MC Check Bank Draft

Credit Card Expiration Date: \_\_\_\_\_

Card #: \_\_\_\_\_

NOTE: Turbo Editor Toolbox and Turbo GameWorks are available for the IBM PC and true-compatibles using Turbo Pascal 3.0 ONLY.

H11 COD's and Purchase Orders will not be accepted by Borland International. California residents add 6% sales tax. Outside USA add \$10 and make payment by bank draft, payable in U.S. dollars drawn on a U.S. bank.